



Anil Kumar Tilanthe

Development of a Learning Analytics Application for a Coding Learning Game

Master's Thesis

to achieve the university degree of

Master of Science

Master's degree programme: Software Engineering and Management

submitted to

Graz University of Technology

Supervisor

Gütl, Christian, Assoc.Prof. Dipl.-Ing. Dr.techn.

Institute of Interactive Systems and Data Science

Co-supervisor

Steinmaurer, Alexander, Mag.rer.nat.

Institute of Interactive Systems and Data Science

Head: Linstaedt, Stefanie, Univ.-Prof. Dipl.-Ing. Dr.

Graz, April 2021



Anil Kumar Tilanthe

Entwicklung einer Lernanalytik-Anwendung für ein Coding-Lernspiel

Master Arbeit

zur Erlangung des akademischen Grades

Diplom-Ingenieur

Masterstudium: Software Entwicklung und Wissenschaft

eingereicht an der

Technische Universität Graz

Betreuer

Gütl, Christian, Assoc.Prof. Dipl.-Ing. Dr.techn.

Institute of Interactive Systems and Data Science

Mitbetreuer

Steinmaurer, Alexander, Mag.rer.nat.

Institute of Interactive Systems and Data Science

Head: Linstaedt, Stefanie, Univ.-Prof. Dipl.-Ing. Dr.

Graz, April 2021

EIDESSTATTLICHE ERKLÄRUNG

AFFIDAVIT

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Datum / Date

Unterschrift / Signature

Acknowledgment

I would like to especially thank Mag.rer.nat. Alexander Steinmaurer for his support, precious inputs, immense patience, motivation, and guidance throughout my thesis. His valuable ideas and insights provided a direction for the entire thesis.

I want to express my deepest gratitude to my supervisor, Assoc. Prof. Dr.techn. Christian Gutl for giving me this opportunity, and for his immense patience towards project completion. His vast experience, guidance, valuable inputs, and support during all project phases guided and motivated me.

Finally, I would like to thank my parents for their support. I thank Herbert, Nitikorn, Max, Harry, Brenda, Marina, and all my friends for being there for me during my ups and downs of my Master's degree and general life.

Abstract

Learning to program can be a challenging task for novice learners. Considering educational games for acquiring coding skills, educators and game developers need to identify students' course progress and analyze their game-related data to provide timely assistance to students in need and improve course content and game design. Analyzing the programming solutions submitted by students who are learning coding skills could help educators understand how students learn and plan and improve course content to enhance learning.

This thesis describes the development and introduction of a learning analytics tool for the mobile learning game sCool. The mobile learning game sCool is an educational game for learning programming or computational skills, providing an immersive and engaging experience to its players. The introduction of additional capabilities of learning analytics for educators and sCool game developers aims to provide them with various insights about student's course progress, performance, strategies undertaken for course completion, the effectiveness of course content for teaching programming concepts, and engagement with the game. The information could assist in analyzing student's performance and improve the course content to meet the needs of students. The information could further assist game developers with insights into student's in-game behavior, engagement, and details of interactions with the game.

An evaluation of the sCool learning analytics application was conducted where the application prototype was tested by 22 participants who tried to gain insights into student's performance and game-related data using the application. The evaluation showed that the participants could analyze student's progress in courses and identify poorly performing students. The participants also were able to identify the effectiveness of course content for teaching programming concepts. It can be concluded that the learning analytics application for coding learning game sCool helps users to gain insights into student's course progress, performance, understanding or use of programming concepts, and in-game behavior.

Kurzfassung

Programmieren zu lernen kann für AnfängerInnen eine schwierige Aufgabe sein. Um das spielerische Erlernen von Programmierfähigkeiten mithilfe von Lernspielen zu ermöglichen, müssen PädagogInnen und SpieleentwicklerInnen den Lernfortschritt verfolgen und die spielbezogenen Daten analysieren können, um SchülerInnen, die Hilfe benötigen, rechtzeitig unterstützen zu können, sowie Kursinhalte und Spieldesign zu verbessern. Durch die Analyse des Codes der SchülerInnen, können PädagogInnen die Lernenden unterstützen, Kursinhalte planen und verbessern, um somit das Lernergebnis zu verbessern.

Diese Arbeit beschreibt die Entwicklung und Einführung eines Lernanalysetools für das mobile Lernspiel sCool. sCool ist ein Spiel zum Erlernen von Programmierfähigkeiten, das seinen SpielerInnen eine immersive und fesselnde Erfahrung bietet. Die Einführung zusätzlicher Möglichkeiten der Lernanalyse für PädagogInnen und SpieleentwicklerInnen soll Einblicke in den Lernfortschritt, die Leistung, das Engagement und die Strategien der SpielerInnen für den Kursabschluss bieten. Die Informationen könnten dabei helfen, die Leistungen der SchülerInnen zu analysieren und den Kursinhalt zu verbessern, um den Bedürfnissen der SchülerInnen gerecht zu werden. Die Informationen können außerdem den SpieleentwicklerInnen helfen, Einblicke in das Verhalten der SpielerInnen im Spiel, das Engagement und die Details der Interaktionen mit dem Spiel zu erhalten.

Es wurde eine Evaluierung der neu entwickelten Lernanalyseplattform durchgeführt, bei der der Prototyp von 22 TeilnehmerInnen getestet wurde, die versuchten, mithilfe der Anwendung Einblicke in die Leistung der SchülerInnen und spielbezogene Daten zu gewinnen. Die Auswertung zeigte, dass die TeilnehmerInnen den Fortschritt der SchülerInnen in den Kursen analysieren und leistungsschwache SchülerInnen identifizieren konnten. Die TeilnehmerInnen waren auch in der Lage, die Effektivität von Kursinhalten für die Vermittlung von Programmierkonzepten zu identifizieren. Es kann gefolgert werden, dass die Analyseplattform den BenutzerInnen hilft Einblicke in den Kursfortschritt, die Leistung, das Verständnis oder die Anwendung von Programmierkonzepten und das Verhalten im Spiel zu gewinnen.

Contents

| | |
|---|-----------|
| Acronyms | xv |
| 1 Introduction | 2 |
| 1.1 Aims and Objectives | 5 |
| 1.2 Methodology and Contribution | 6 |
| 1.3 Structure | 7 |
| 2 Background and Related Work | 9 |
| 2.1 Serious Games | 9 |
| 2.1.1 Learning Mechanics and Game Mechanics | 10 |
| 2.1.2 Game Metrics | 11 |
| 2.1.3 Coding Constructs | 14 |
| 2.1.4 Serious Games in Computer Science education | 16 |
| 2.1.4.1 CodeMonkey | 16 |
| 2.1.4.2 Ozaria | 17 |
| 2.1.4.3 sCool | 18 |
| 2.1.5 Overview | 20 |
| 2.2 Learning Analytics | 21 |
| 2.2.1 Learning Analytics for Serious Games Architecture | 22 |
| 2.2.2 Learning Analytics Information Model | 23 |
| 2.2.3 Code Analysis | 25 |
| 2.2.4 Learning Analytics Tools and Dashboards | 27 |
| 2.2.4.1 CodeMonkey | 28 |

CONTENTS

| | | |
|----------|--|-----------|
| 2.2.4.2 | Ozaria | 31 |
| 2.2.4.3 | sCool | 34 |
| 2.2.5 | Overview | 35 |
| 2.3 | Data Visualization | 36 |
| 2.3.1 | Visualization techniques | 37 |
| 2.3.2 | Visualization Tools and Frameworks | 40 |
| 2.3.2.1 | Tableau | 41 |
| 2.3.2.2 | Infogram | 41 |
| 2.3.2.3 | RAW Graphs | 42 |
| 2.3.2.4 | D3.js | 43 |
| 2.3.2.5 | Chart.js | 44 |
| 2.3.2.6 | Plotly | 45 |
| 2.3.3 | Overview | 46 |
| 2.4 | Summary | 47 |
| 3 | sCool | 48 |
| 3.1 | Game Design | 48 |
| 3.2 | System Architecture | 49 |
| 3.3 | Mobile Game | 50 |
| 3.4 | Web Platform | 54 |
| 3.5 | Summary | 57 |
| 4 | Requirements | 58 |
| 4.1 | Additional Features | 58 |
| 4.2 | User Groups | 59 |
| 4.3 | Requirements | 59 |
| 4.3.1 | Functional Requirements | 60 |
| 4.3.2 | Non-Functional Requirements | 62 |
| 4.3.3 | Data | 63 |
| 4.4 | Conceptual Architecture | 63 |
| 4.5 | Summary | 64 |

CONTENTS

| | | |
|----------|--|------------|
| 5 | Development | 66 |
| 5.1 | Architecture | 66 |
| 5.2 | Implementation | 68 |
| 5.2.1 | Connecting with Database | 69 |
| 5.2.2 | Python Code Parser for Code Concepts Used | 69 |
| 5.2.3 | JSON Parser | 71 |
| 5.2.4 | Data Processing | 72 |
| 5.2.5 | Login Redirect and User Session Management | 75 |
| 5.2.6 | Model-View-Controller | 77 |
| 5.2.7 | Data Visualization Techniques | 79 |
| 5.3 | User Interface | 81 |
| 5.4 | Features | 83 |
| 5.4.1 | Educator User Group | 83 |
| 5.4.1.1 | Learning Activity/Class Details Section | 83 |
| 5.4.1.2 | Student Details Section | 84 |
| 5.4.1.3 | Custom Details Section | 89 |
| 5.4.2 | Administrator User Group | 95 |
| 5.4.2.1 | Game Data | 95 |
| 5.4.2.2 | Learning activities comparison | 95 |
| 5.5 | Summary | 99 |
| 6 | Evaluation | 101 |
| 6.1 | Scope | 101 |
| 6.2 | Instruments and Setup | 102 |
| 6.2.1 | System Usability Scale (SUS) | 104 |
| 6.2.2 | NASA Task Load Index (NASA-TLX) | 106 |
| 6.2.3 | Application-related questions | 108 |
| 6.3 | Participants | 108 |
| 6.4 | Evaluation and Results | 112 |
| 6.4.1 | Task Completion and Scores | 112 |

CONTENTS

| | | |
|----------|--|------------|
| 6.4.2 | Evaluation Responses of Participants for Application Specific Question | 117 |
| 6.4.2.1 | Educator Participants | 117 |
| 6.4.2.2 | Participants Excluding Educators | 123 |
| 6.4.3 | System Usability Scale (SUS) | 123 |
| 6.4.4 | NASA-TLX | 127 |
| 6.4.5 | Suggestions and Feedback | 129 |
| 6.4.5.1 | Educator Participants | 129 |
| 6.4.5.2 | Participants Excluding Educators | 129 |
| 6.5 | Discussion and Limitations | 131 |
| 7 | Lesson Learned | 133 |
| 7.1 | Literature | 133 |
| 7.2 | Development | 134 |
| 7.3 | Outcome | 135 |
| 8 | Conclusion | 137 |
| 8.1 | Conclusion | 137 |
| 8.2 | Future Work | 138 |
| | Bibliography | 140 |
| A | Questionnaire | 151 |
| A1 | Intro | 151 |
| A2 | Tasks | 153 |
| A3 | Application Specific Questions | 157 |
| A4 | NASA-TLX | 160 |
| A5 | System Usability Scale | 161 |

List of Figures

| | | |
|------|---|----|
| 2.1 | CodeMonkey block coding course | 17 |
| 2.2 | CodeCombat Ozaria players game view: players control an avatar using code | 18 |
| 2.3 | sCool mobile game: view of practical mode where players control robot to support the crew in repairs of their space shuttle (sCool version 3, year 2020; Mosquera et al., 2020; Steinmaurer et al., 2020; Steinmaurer et al., 2019) | 19 |
| 2.4 | sCool Web application: overview of existing courses and new course create button (sCool version 3, year 2020; Steinmaurer et al., 2020; Steinmaurer et al., 2019; Mosquera et al., 2020;) | 20 |
| 2.5 | Overview of learning analytics for serious games architecture (after Alonso-Fernandez et al., 2017; Freire et al., 2016) | 22 |
| 2.6 | Model of users processing information from learning analytics applications (after Verbert et al., 2013; Li et al., 2010) | 24 |
| 2.7 | Structure of AST tree generated of code statement from listing 2.1 | 27 |
| 2.8 | CodeMonkey dashboard for teachers showing students progress in the course | 29 |
| 2.9 | CodeMonkey dashboard for teachers showing students overall grades | 29 |
| 2.10 | CodeMonkey dashboard for teachers showing students proficiency in computer science topics | 30 |
| 2.11 | CodeMonkey dashboard for students showing progress in current course and available courses | 31 |

LIST OF FIGURES

| | | |
|------|--|----|
| 2.12 | Ozaria dashboard for teachers class view depicting students progress in the course and other features | 32 |
| 2.13 | Ozaria dashboard for teachers class view depicting code submissions of students | 33 |
| 2.14 | Ozaria dashboard for students depicting joined classes and progress in current class | 34 |
| 2.15 | Rock paintings from the Cave of Beasts (Gilf Kebir, Libyan Desert) Estimated 7000 BP (Schmillen, 2014) | 37 |
| 2.16 | Data visualization techniques | 39 |
| 2.17 | Tableau Covid-19 global tracker dashboard (tableau, 2021a) | 41 |
| 2.18 | Infogram template of a sample dashboard (Infogram, 2021b) | 42 |
| 2.19 | RAW Graph movie visualization of production budget vs box office (graph, 2021) | 43 |
| 2.20 | D3.js example bubble map : county population in USA (Bostock, 2014) | 44 |
| 2.21 | Chart.js example bubble chart (Bostock, 2021) | 45 |
| 2.22 | Plotly example mixed subplots from documentations (Plotly, 2021) | 46 |
| 3.1 | sCool - System Architecture (adapted after Steinmaurer et al., 2020; Steinmaurer, 2019) | 50 |
| 3.2 | sCool Concept learning. Players find disks guarded by enemies. (sCool version 3, year 2020.) | 51 |
| 3.3 | sCool Concept learning. When Players retrieve disks they are introduced to new programming concepts. (sCool version 3, year 2020.) | 52 |
| 3.4 | sCool Concept learning. Introduction to new programming concepts is followed by related questions. (sCool version 3, year 2020.) | 52 |
| 3.5 | sCool Practical learning. Robot in a 2-dimensional grid with a disk to reach. (sCool version 3, year 2020.) | 53 |
| 3.6 | sCool Practical learning. Code interface with code blocks and other options. (sCool version 3, year 2020.) | 54 |

LIST OF FIGURES

| | | |
|------|---|----|
| 3.7 | sCool Hierarchical Course Tree (redrawn after Steinmaurer, 2019; Steinmaurer et al., 2020) | 55 |
| 3.8 | sCool Web application: overview of existing courses and new course create button (sCool version 3, year 2020.) | 56 |
| 3.9 | sCool Web application: Overview of Concept learning mode tasks of a course-skill (sCool version 3, year 2020.) | 56 |
| 3.10 | sCool Web application: Overview of Practice mode tasks of a course-skill (sCool version 3, year 2020.) | 57 |
| 4.1 | sCool - system architecture with the integration of the sCool learning analytics application. (adapted after Steinmaurer et al., 2019; Steinmaurer et al., 2020; Steinmaurer, 2019) | 64 |
| 5.1 | Simplified Architecture | 68 |
| 5.2 | Steps of data processing | 73 |
| 5.3 | Overview of user interface | 82 |
| 5.4 | Overview of details section | 85 |
| 5.5 | Overview of students section | 86 |
| 5.6 | Overview of students game interactions | 87 |
| 5.7 | Overview of students game timeline chart with mouseover user interaction | 88 |
| 5.8 | Overview of Custom section form | 90 |
| 5.9 | Example of a custom plot | 91 |
| 5.10 | A custom plot of the number of errors that occurred in program solutions submitted by students for course tasks showing that students had many errors in a particular practice task | 92 |
| 5.11 | A custom plot of the number of errors that occurred in players code submissions | 93 |
| 5.12 | A custom plot of the number of times players switched to read description for each of the tasks in a sample class | 94 |
| 5.13 | Overview of students section | 96 |
| 5.14 | Overview of game data section | 97 |

LIST OF FIGURES

| | | |
|------|---|-----|
| 5.15 | Overview of learning activities comparison section | 98 |
| 6.1 | Gender of participants | 110 |
| 6.2 | Age of participants (in years) | 110 |
| 6.3 | Profession of participants | 111 |
| 6.4 | Highest educational degree of participants | 111 |
| 6.5 | Familiarity of participants with e-learning tools or learning software . | 112 |
| 6.6 | Distribution of evaluation scores of all participants for task completion | 113 |
| 6.7 | Evaluation scores of participants for task completion | 114 |
| 6.8 | Able to identify course progress of students | 118 |
| 6.9 | Able to identify students performing poorly | 118 |
| 6.10 | Able to identify tasks which were hard or easy | 119 |
| 6.11 | Would recommend application to a colleague | 119 |
| 6.12 | (Educators) Found the tool meaningful | 121 |
| 6.13 | (Educators) Found the data collected and analyzed relevant for Edu- cation purpose or useful for analyzing students progress | 122 |
| 6.14 | (Educators) Would like to use the application for analyzing students . | 122 |
| 6.15 | Results of SUS scores of each participants in the evaluation | 124 |
| A.1 | Participants intro | 151 |
| A.2 | Participants familiarity with e-learning tools | 152 |
| A.3 | Intro pre-recorded audio-video | 152 |
| A.4 | Task 1 | 153 |
| A.5 | Task 2 | 153 |
| A.6 | Task 3 | 153 |
| A.7 | Task 4 | 154 |
| A.8 | Task 5 | 154 |
| A.9 | Task 6 | 155 |
| A.10 | Task 7 | 155 |
| A.11 | Task 8 | 155 |
| A.12 | Task 9 | 156 |
| A.13 | Task 10 | 156 |

LIST OF FIGURES

| | |
|--|-----|
| A.14 Task 11 | 156 |
| A.15 Participants educators I | 157 |
| A.16 Participants educators II | 158 |
| A.17 Participants except educators | 159 |
| A.18 NASA-TLX | 160 |

List of Tables

| | | |
|-----|--|-----|
| 2.1 | Overview of serious games for computational skill learning | 20 |
| 2.2 | Overview of learning analytics tools and dashboards for educational games | 36 |
| 6.1 | Overview of the evaluation tasks and task subgroups | 104 |
| 6.2 | Overview of the complete curved grading scale with range of SUS scores and corresponding grade and percentile range (based on Lewis and Sauro, 2018) | 106 |
| 6.3 | Application-related questions | 108 |
| 6.4 | Overview of the task wise answers | 116 |
| 6.5 | Overview of the responses of application-specific questions | 120 |
| 6.6 | Overview of the responses on SUS | 126 |
| 6.7 | NASA-RTLX results | 128 |
| A.1 | System Usability Scale | 161 |

Listings

| | | |
|-----|---|----|
| 2.1 | A simple python program with an assignment statement | 26 |
| 5.1 | Connecting to database using pyodbc | 69 |
| 5.2 | Parsing a python program to create AST for examining coding concepts used | 70 |
| 5.3 | Parsing a JSON string | 71 |
| 5.4 | Players game-related data processing step | 74 |
| 5.5 | User redirect form | 75 |
| 5.6 | User login POST request handler | 76 |
| 5.7 | Model-View-Controller | 77 |
| 5.8 | Data visualization highlighting key numbers | 79 |
| 5.9 | Data visualization showing players data using a horizontal bar chart . | 80 |

Acronyms

API Application Programming Interface. 49

AST Abstract Syntax Tree. xiv, 26, 69, 71

COVID-19 Coronavirus disease 2019. 2, 102, 108, 137

CSS Cascading Style Sheets. 43, 44, 79, 139

CSTA Computer Science Teachers Association. 17

DAU Daily Active Users. 11, 12

GCP Google Cloud Platform. 135

HTML HyperText Markup Language. 79

HTML5 HyperText Markup Language. 43, 44

HTTP Hypertext Transfer Protocol. 75

ISTE International Society for Technology in Education. 17

JSON JavaScript Object Notation. xiv, 49, 71

MAU Monthly Active Users. 11, 12

MAUU Monthly Average Unique Users. 12

MCQ Multiple Choice Question. 103

MMOG Massively multiplayer online game. 12

MVC Model–view–controller. 67, 77

NASA National Aeronautics and Space Administration. 104

NASA-RTLX NASA Raw Task Load Index. 107, 132

NASA-TLX NASA Task Load Index. 106, 107, 132, 138

NETP National Education Technology Plan. 3

ODBC Open Database Connectivity. 69

OS Operating System. 68

PCU Peak Concurrent Users. 12

REST Representational State Transfer. 49, 134

SUS System Usability Scale. xiii, 104–106, 123–125, 138

SVG Scalable Vector Graphics. 43

UI User Interface. 66, 67, 81, 83, 99, 135, 138

Chapter 1

Introduction

The recent Coronavirus disease 2019 (COVID-19) pandemic has highlighted the importance of distance learning. The pandemic has caused widespread school closures in 185 countries, and up to 89.4% of learners were forced to stay home at a certain point in time (UNESCO, 2021). The pandemic forcing students to learn from home has emphasized the importance of effective distance learning. Learning games or serious games are one of the mediums of distance learning (Bates, 1997; Bidarra, 2009). This thesis will focus on learning analytics for educational games or serious games.

Games provide engaging and motivational content that connects with players more deeply than linear forms of media, and that presents scenarios where players are challenged (rather than forced) to perform better. Video gaming is a prevalent activity among adults as 43% of adults say they play video games on a computer, TV, console, or cellphone (Pew Research Center, 2018). Puzzle and strategy games are the most popular genre among the players as around 62% of players play them. The amount of time spent playing video games by people has also increased over the years as, on average, people played games for 8.45 hours per week in 2021 (*State of Online Gaming 2021*, 2021) compared to an average of 6.20 hours per week in 2020 and 5.96 hours each week in 2018 (*The State of Online Gaming – 2020*, 2020; *The State Of Online Gaming – 2018*, 2018).

Players spend time in games because of the engaging and motivational content

in a virtual environment that players can relate to more deeply than a linear form of media, and that has scenarios where players feel challenged to perform better rather than being forced. This motivated interest in learning games or games for educational purposes. A good game design is closely aligned with an excellent educational experience, and games can serve as fundamental and powerful learning tools (Koster, 2004). It is one thing to read about city management and another to be a mayor of a virtual city, managing it and learning necessary finance and management concepts in a game, which can be considered a type of learning strategy within the *experiential learning* approach (Patino et al., 2016).

As described in the U.S. Department of Education's National Education Technology Plan (NETP) (Department of Education, 2010a), increasing use of online learning offers opportunities to integrate learning and assessment:

"The same technology that supports learning activities gathers data in the course of learning that can be used for assessment. [...] An online system can collect much more and much more detailed information about how students are learning than manual methods. As students work, the system can capture their inputs and collect evidence of their problem-solving sequences, knowledge, and strategy use, as reflected by the information each student selects or inputs, the number of attempts the student makes, the number of hints and feedback given, and the time allocation across parts of the problem" (Department of Education, 2010a, p. 30).

Learning dashboards integrate information from learning tools and other relevant sources to provide a comprehensive visual representation of student's progress. One of the recommendations from the U.S. Department of Education's National Education Technology Plan (NETP) (Department of Education, 2017) is:

"States, districts, and others should design, develop, and implement learning dashboards, response systems, and communication pathways that give students, educators, families, and other stakeholders timely and actionable feedback about student learning to improve achievement and instructional practices" (Department of Education, 2017, p. 67).

The feedback based on learning dashboard insights ensures that relevant data inform decisions about learning and course content. This process relies on learning systems collecting, aggregating, and analyzing large amounts of data. The learning dashboard, providing learning analytics capabilities, helps discover student's interactions and performance, gain feedback, and improve course content to achieve effective learning (Long and Siemens, 2011). For educators, the availability of insight into learner's performance and interactions with the learning system can significantly help plan and improve course content and activities. The insights from learning analytics could also assist educators to identify students who are performing poorly and may require assistance or other approaches to enhance learning (Dietz-Uhler and Hurn, 2013; Khalil and Ebner, 2015). Overall, learning analytics can help to improve the quality and value of the learning experience (Long and Siemens, 2011).

This thesis focuses on educational tools for acquiring programming or computational skills. The mobile learning game sCool is an educational game for acquiring coding or computational skills. sCool was initially developed in 2017 in a cooperation between Graz University of Technology and Westminster University (A. Kojic, 2017; Steinmaurer, 2019). sCool is also continuously improved with new features and improvements in usability and learning effectiveness by various collaborators. sCool provides an immersive and engaging experience to its players, which follows the storyline of an escape of a space shuttle and its crew members, finding themselves lost in space and crashed on an alien planet. To escape the alien planet, the players must learn programming concepts and collect items to repair their space shuttle by controlling and moving a robot using code.

The sCool system consists of two components: a mobile video game for students to play and learn and a web application to create course content for educators. The learning approach of sCool is divided into two parts: an initial concept-learning part and a practical programming learning part. In the concept learning part of the game, the students learn programming concepts, and in the practical part, they apply the theoretical concepts using the Python programming language where they control a robot avatar using Python code to collect disks and items for their space shuttle (Steinmaurer, 2019). There are draggable code blocks that get converted to editable

Python commands, which the robot can execute. A virtual keyboard is available to write or modify code (Steinmaurer, 2019).

A novice programming learner, regardless of age, could face many difficulties while attempting to understand and learn programming concepts such as classes, objects, variables, arrays, and loops (Lahtinen et al., 2005). These difficulties have to be acknowledged and recognized in order to be able to assist the students. Analyzing programming solutions submitted by students could also help educators understand how novice programmers learn and how to teach them best (Albluwi and Salter, 2020). Additionally, it is critical to bring young children in contact with basic programming concepts in a developmentally appropriate manner and provide timely assistance to students in need (Kanaki and Kalogiannakis, 2018).

1.1 Aims and Objectives

The primary purpose of this thesis is to provide learning analytics capabilities to key users involving educators and game developers of the mobile learning game sCool. The project intends to create an informative tool providing various insights about student's course progress, performance, and interactions in the game extracted from the collected game-related data of students. The thesis focuses on providing an informative tool rather than a decision-making tool (Dillenbourg et al., 2011).

This thesis can be divided into theoretical and practical parts. The theoretical part investigates various concepts related to educational games for learning coding or computational skills and concepts related to learning analytics on players data of such educational games. This investigation is conducted through three stages comprising of:

- Background and exploration of educational games for learning coding or computational skills and related concepts.
- Learning analytics for educational games for learning coding or computational skills and related concepts.

- Data visualization techniques, tools, and related concepts for communicating knowledge effectively.

Creating a web-based application is considered to provide educators and game developers learning analytics capabilities with the tool providing platform-independent multi-user support. The application should extract meaningful information from player's game-related data and present it in an understandable format. The information presented must be meaningful and easy to comprehend.

A sCool learning analytics tool was developed in order to fulfill these goals. The practical part of the thesis is divided into two main phases where this process is presented:

- Development of sCool learning analytics tool.
- Evaluation of the sCool learning analytics tool.

The sCool learning analytics tool should assist educators and game developers with information such as students' progress in courses, strategies undertaken by students for course completion, the effectiveness of tasks and course content in teaching programming concepts to students, and students' engagement with the game. Such information should assist users in identifying any students who are performing poorly and requires assistance or incentives for performance improvement. The information should assist educators and game developers in improving course content to meet the needs of students. Furthermore, the usability of the tool should be analyzed.

1.2 Methodology and Contribution

This research is based on the previous work of sCool developers who developed the application and have kept releasing new features to support the vibrant and engaging learning platform and improving its usability (A. Kojic, 2017; M. Kojic, 2017; Steinmaurer, 2019). The sCool system consists of a mobile coding learning game and a web application. The web application provides the functionality of course content creation for educators and game developers. The sCool game currently

provides an overview of course progress to students within the mobile game. Further requirements of providing learning analytics capabilities to educators and sCool game developers were considered for this project.

The result of the project should be a creation of an informative tool providing critical insights regarding the performance of students in courses on the sCool platform to its users, rather than a decision-making tool. The introduced application should assist educators and game developers with information such as students' progress in courses, various strategies undertaken by students for course completion, the effectiveness of tasks in courses for teaching programming concepts, and students' engagement with the game. Such information should assist the users to identify any students who are performing poorly and may require assistance or incentives for performance improvement and further improve the course content. The programming solutions submitted, various in-game actions of students, game strategies undertaken to complete the courses on sCool, progress in courses, effectiveness and completion rate of tasks, and recurring programming errors faced by students should also be considered for analysis. A platform-independent application with multi-user support such as a web application and its availability, usability, and ease of communicating information using various data visualization techniques should be considered.

1.3 Structure

This thesis consists of eight chapters that describe different phases of the project. Chapter 2 represents the theoretical part of the thesis. In this chapter, educational games and related concepts are described using educational games for learning programming or computational skills. The chapter also describes learning analytics and related concepts followed by examples of learning analytics tools for various educational games for learning programming or computational skills. Additionally, the chapter introduces data visualization techniques followed by examples of data visualization tools and frameworks. Chapter 3 is regarding the mobile video game sCool. It describes the mobile game and its environment, explaining the architecture, components, and game types in sCool. In chapter 4, the requirements and concepts for

a learning analytics tool for sCool are introduced. Chapter 5 covers technical specifications and the implementation of the sCool learning analytics tool. In chapter 6, the evaluation of the tool is presented, and the related research questions for the thesis are analyzed. Chapters 7 and 8 summarize the learning lessons and provide suggestions for future development of learning analytics capabilities on sCool.

Chapter 2

Background and Related Work

In this chapter, the background and related scope of the thesis is established. Important concepts are introduced and described. The chapter describes concepts related to educational games followed by educational games for teaching coding or computation skills. The chapter also describes learning analytics and related concepts followed by examples of learning analytics tools for various educational games for learning programming or computational skills. Additionally, the chapter introduces data visualization techniques followed by examples of data visualization tools and frameworks.

2.1 Serious Games

The term serious games refer to "*games that do not have entertainment, enjoyment or fun as their primary purpose*" (Michael and Chen, 2006). Games have been used for educational purposes for quite some time. These games also include non-digital games. However, by the term serious games, most academics refer to the use of digital games for educational purposes (Djaouti et al., 2011; Abt, 1981).

Serious games also produce a tremendous amount of player's game-related data, which is collected and stored for further analysis. Meaningful information can be extracted from the collected data and could be used to improve the game itself and

to provide better learning experiences or learning process to players in the future. Serious game research fields also include learning science, psychology, and computer sciences (Patino et al., 2016).

2.1.1 Learning Mechanics and Game Mechanics

The serious game mechanics are the game elements/aspects which link a pedagogical practice to game mechanics. Various game mechanism can be mapped to human learning mechanism providing an understanding concerning which game mechanics complement learning (Arnab et al., 2014). The pedagogical and game elements in a serious game can identify learning and entertainment features and their interrelations in a game.

Some of the learning mechanics and game mechanics characteristics are (Arnab et al., 2014; Lim et al., 2013):

- *Learning Mechanics*
 - **Action/Task:** learners perform tasks in order to get rewards.
 - **Instructional:** learners follow instructions to learn concepts or perform tasks.
 - **Plan:** learners have to make a strategic plan in order to solve problems.
 - **Simulation:** learners gain role playing experiences.
 - **Feedback:** oral or written development advice to learners on their performance.
 - **Explore:** encouraging learners to explore and experiment for teaching generalised thinking and problem solving skills.
 - **Experimentation:** involving laboratory or practical learning.
 - **Analyse:** analysis and diagnostics to identify weakness or strengths.
 - **Competition:** competitive learning to foster creativity and problem-solving skills and reaching an optimal position.

- **Motivation:** using marks or grades to motivate students to achieve their best potential.
- *Game Mechanics*
 - **Resource Management:** players make decisions based on resource and time constraints.
 - **Rewards:** rewards are designed as incentives to keep players motivated and engaged.
 - **Capture/Eliminate:** players gain points by capturing or eliminating targets.
 - **Questions and Answers:** as means of engaging and interacting with players.
 - **Role Playing:** players play to develop their role as virtual characters.
 - **Collecting:** players collect elements of knowledge, skill, competencies or rewards represented as virtual objects.
 - **Cascading Information:** information available in minimum chunks relative to an appropriate level of understanding.

2.1.2 Game Metrics

Game Metrics are various interpretable performance measures, such as the number of daily active users, the average completion time of users over multiple game levels, revenue per day, and average session duration of users (Drachen et al., 2013a). Metrics can be individual features, complex aggregate or calculated values from multiple features. These performance measures are important to gain knowledge of the player's accomplishments after performing several game tasks. Game metrics such as Daily Active Users (DAU) and Monthly Active Users (MAU) are used as a way of measuring user engagement (Junaidi et al., 2018; Hui, 2013). The DAU and MAU data can also provide retention rates as a game's ability to retain its users (Hui, 2013).

Game metrics can be classified into following categories (Mellon, 2009; Junaidi et al., 2018; Drachen et al., 2013b):

- **User metrics:** User metrics are the metrics related to the users who play games. Some of the user metrics are:
 - **Daily Active Users (DAU):** Daily Active Users (DAU) is the number of unique users per day (Fields, 2013). It is usually calculated over a period of the last seven days. Actions such as the user’s brief visit to the game could also count towards DAU.
 - **Monthly Active Users (MAU):** Monthly Active Users (MAU) is the number of users in a given calendar month (Fields, 2013). MAU is usually calculated from the first to the last day of a month. The game developers should clarify if they are attempting to include unique users only or not. Measuring unique users can be termed as Monthly Average Unique Users (MAUU).
 - **Engagement:** Engagement or user engagement is a measure of how invested the users are in playing a game. Time spent playing the game by users provides a measure of user engagement. Most of the games measure engagement in minutes and seconds. Since players can also wait idle, the total number of user inputs such as keystrokes, inputs, or clicks could also provide a measure of engagement in a given session.
 - **Peak Concurrent Users (PCU):** For games such as Massively multi-player online game (MMOG), it is essential to plan for peak simultaneous users. This metric of concurrent users at the peak is referred to as the game’s Peak Concurrent Users (PCU) (Fields, 2013). Poor planning for PCU could result in long wait times for users in queue or server crashes.
 - **Retention Rate:** Retention rate is a measure of the number of players who are returning to play the game after a certain period from their first experience with the game. Day 1 retention rate provides information about how many players played the game again after one day from

their first experience with the game. Similarly, the retention rate can be measured for other time periods. The retention rate can be increased with solutions such as the release of new features, innovations, and regular updates.

- **Process metrics:** Process metrics are the metrics related to the actual process of developing games. These include game development methodology and employee's effectivity and efficiency. Game development is a creative process. The methodologies such as agile methodology, waterfall, or scrum must be analyzed. As a creative process, it has necessitated the use of agile development methods. The composition of the game's developer team is also essential. The measure of the team having the right people in the correct position must be checked.
- **Performance metrics:** Performance metrics are the metrics related to the performance of the technical and software infrastructure behind the game. The number of bugs found per hour, day, week, or another time frame, is an example of performance metrics. Performance metrics are used in quality assurance to monitor the health of the game.

One of the vital pieces of information is to find if some of the game levels are incredibly challenging or remarkably easy (Junaidi et al., 2018; Freire et al., 2016). An incredible challenging task or game level could act as a stumbling point, which hinders a player's progress from furthering game levels. This knowledge could help game developers and content creators in order to improve game design and content. Another focus is on identifying popular game levels where players spend more time and unreachable game areas that players never visit.

The metric of time taken to solve game tasks or game levels could also provide an assessment of player's cognitive flexibility (Plass et al., 2013). Players with higher cognitive flexibility for a task are likely to solve a game task faster and use diverse solutions. The time logs of various operations performed by the players also provide information about their game strategies. Different players could use different strategies in order to achieve the highest scores in a learning environment.

Game metrics also support learning assessment by providing essential indications of a training's success (Kiili et al., 2018). The game data could be used to demonstrated achievement of learning goals and objectives of the serious game (Bellotti et al., 2013). Evaluating learning in the game involves measuring the increase in knowledge before and after playing the game (Bachvarova et al., 2012). Completion and grades are key measures in educational contexts (Freire et al., 2016). Some of the learning and performance indicators for educational objectives are (Plass et al., 2013):

- Total score
- Number of tasks solved
- Number of levels completed
- Number of game resources accessed
- Success and failures within a certain time frame

2.1.3 Coding Constructs

Computational thinking is increasingly expected to be considered as a fundamental skill for everyone in 21st century (Wing, 2006; Grover and Pea, 2013; Barr and Stephenson, 2011; Kanaki and Kalogiannakis, 2018). Students also develop computational and critical thinking during the process of learning to code. A novice programming learner, regardless of age, could face many difficulties while attempting to understand and learn programming concepts such as classes, objects, variables, arrays, and loops (Lahtinen et al., 2005). These difficulties have to be acknowledged and recognized in order to be able to assist the students. Some of the most difficult programming concepts could be a recursion, pointers and references, error handling, and using the language libraries (Lahtinen et al., 2005). In addition to programming concepts, tasks such as finding bugs in one's own programs and designing a new program are also perceived as complex and difficult (Lahtinen et al., 2005).

Hence, it is critical to bring young children in contact with basic programming concepts in a developmentally appropriate manner for future development (Kanaki and Kalogiannakis, 2018).

Objects are everywhere. Children start to recognize objects around them from a young age. Children also learn to associate real-world objects with their respective attributes and properties. The basis of object-oriented programming is real-world objects (Cox, 1986; Stefik and Bobrow, 1985). The children could gradually be exposed to object-oriented programming's fundamental principles and made acquainted with programming concepts such as classes, objects, attributes, and variables. Educational games could provide structured training exercises for various coding concepts and separate the training exercises into levels. Once a student completes a level as proof of having gained a basic understanding of the intended concept, the student could learn additional coding constructs in subsequent game levels or practice the learned concepts further. Besides, procedural programming concepts could also be taught.

An understanding of various programming concepts such as variables, loops, arrays, classes, objects, functions, and conditional statements, could be gradually provided to students. Rogozhkina and Kushnirenko (2011) evaluated an open source environment, PiktoMir¹, for teaching programming to children where children instruct a robot to perform various tasks. In their achievement test, where participants completed various programming tasks, results identified that 80% of the 35 children participants aged between 6.5 to 7 years were able to pass the test about loops successfully, whereas almost all of the 41-42 participants aged seven years or less were able to understand how to write simple linear programs where students moved a robot guided with programming instructions.

Besides teaching programming concepts, it is equally important to monitor and identify the student's progress in learning the programming concepts. This vital knowledge helps teachers assist the students who are facing difficulties grasping programming concepts. Teacher's assistance could help the student who may have low motivation to complete tasks by providing incentives towards course progress (Borah,

¹<https://piktomir.ru/>

2013; Muppudathi, 2014) and to be able to successfully understand the programming concepts eventually.

2.1.4 Serious Games in Computer Science education

There are many serious games for learning computational skills. They could be classified into three main categories based on their goals as (Comb  fis et al., 2016):

- **Learning to Code:** This category is of games whose goal is to make users learn to code. Some of the main coding activities include writing code to complete tasks and fixing broken code. Once the code is submitted, the system provides feedback with passed or failed results.
- **Learning Algorithmic thinking:** In this category of games, the focus is not on learning a particular programming language but rather algorithmic thinking through interactive problems and programming concepts.
- **Learning to Create Games:** In this category of games, the users have the possibility to create their own games. The focus is on creativity and design skills.

Below some example of serious games for teaching coding or computational skills are discussed. Their key features are also presented.

2.1.4.1 CodeMonkey

CodeMonkey² is a apps and web-based educational game where kids learn to code (CodeMonkey, 2021). CodeMonkey offers various courses for students of varying experience levels where kids without any prior coding experience could also learn to code. CodeMonkey has pre-created, and ready-to-use courses. The majority of CodeMonkey’s courses also do not require any prior coding experience to teach. The courses are designed for school, extra-curricular, and home use. CodeMonkey covers text-based coding languages of CoffeeScript and Python. CodeMonkey also has

²<https://app.codemonkey.com/>

code-block-based courses, where players can drag-and-drop blocks of code to control an avatar. CodeMonkey’s courses cover concepts such as objects, function calls, arguments, variables, arrays, for loops, function definitions, and loops (CodeMonkey, 2021). Figure 2.1 shows CodeMonkey’s block coding course where the players can drag-and-drop block of code instructions to move the avatar to collect items or to reach a goal location. CodeMonkey also has a dashboard for students where they see available courses and progress in respective courses.

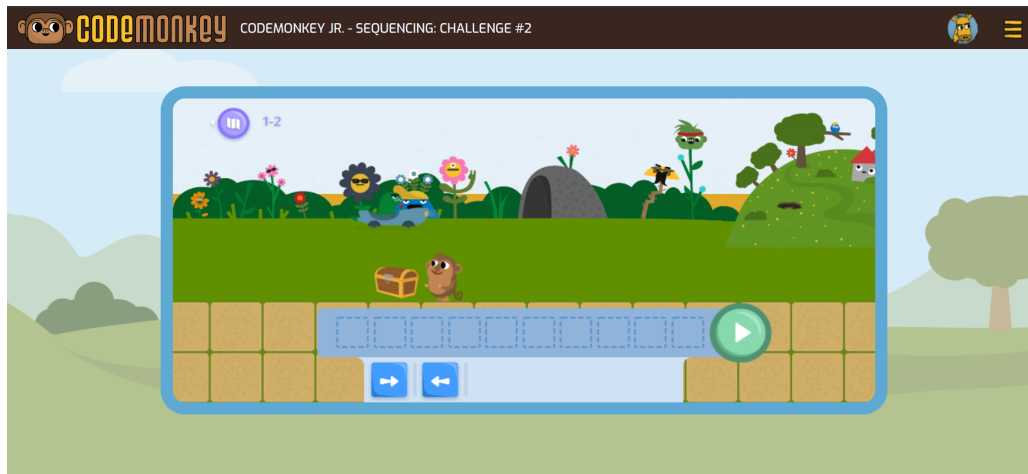


Figure 2.1: CodeMonkey block coding course

2.1.4.2 Ozaria

Ozaria³ is a web-based serious game for learning to code. Ozaria is an immersive story-based fantasy learning environment, where teachers with no coding experience can also teach a full Computer Science curriculum (Ozaria, 2021). Ozaria is aligned to meet CSTA (Computer Science Teachers Association, 2021) and ISTE (International Society for Technology in Education, 2021) K-12 middle school Computer Science education standards. The courses are taught in JavaScript and Python programming languages. Ozaria is designed for both in-person and remote learning settings (Ozaria, 2021). In Ozaria, the players control an avatar using code to fulfill

³<https://www.ozaria.com/>

tasks. Figure 2.2 shows Ozaria’s game view where the players control an avatar using code. The players are provided audio and textual hints to help them understand, learn, and complete the tasks on various levels. Ozaria also provides a dashboard for students where they can see their courses and progress in the respective courses.

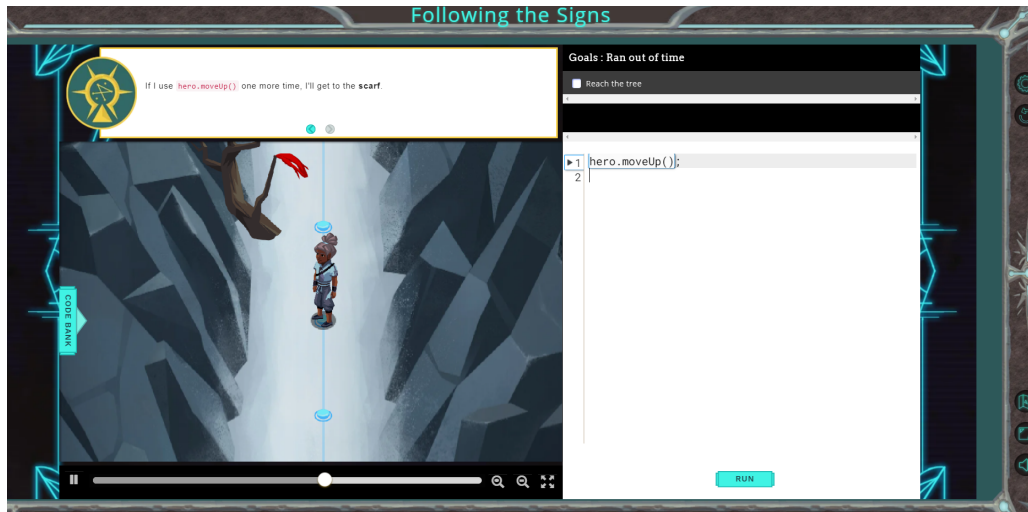


Figure 2.2: CodeCombat Ozaria players game view: players control an avatar using code

2.1.4.3 sCool

The mobile learning game sCool was initially developed in 2017 in a cooperation between Graz University of Technology and Westminster University (A. Kojic, 2017; M. Kojic, 2017; A. Kojic et al., 2018; Steinmaurer et al., 2019; Mosquera et al., 2020). It is a game-based learning tool for computational skills (Steinmaurer et al., 2020). The game narrates the story of a space mission where the shuttle crashed on a foreign planet. The players in a space team member’s role must support the crew in repairs and escape from the planet. The sCool environment consists of two components: a) sCool mobile game and b) web application (Steinmaurer et al., 2020).

The sCool mobile game consists of two game modes: i) concept-learning mode and ii) practical mode. In the first concept-learning mode, the players have to find and collect disks that represent concepts. Alien enemies guard the disks. Players

must avoid or defeat the alien guards to collect disks. On successfully collecting disks, players are presented with concept information followed by a related question. In the practical mode, the players apply previously learned theoretical concepts. The players control a robot remotely. The goal is to collect a disk using the robot and solve other assigned tasks. The robot is controlled by instructions that represent commands in the Python programming language. There are draggable code blocks that get converted to editable Python commands, which the robot can execute (Steinmaurer, 2019; Mosquera et al., 2020). A virtual keyboard is available to write or modify code. Figure 2.3 shows sCool’s practical game mode where players control the robot with code to collect disks and fulfill task requirements.

The web application supports the adaptive learning content. Educators can create new learning content for their courses as well as modify and update them when required. Educators can define learning content for both concept-learning and practical missions (Steinmaurer, 2019). Figure 2.4 shows sCool’s web application showing an overview of existing courses and a new course create button.



Figure 2.3: sCool mobile game: view of practical mode where players control robot to support the crew in repairs of their space shuttle (sCool version 3, year 2020; Mosquera et al., 2020; Steinmaurer et al., 2020; Steinmaurer et al., 2019)

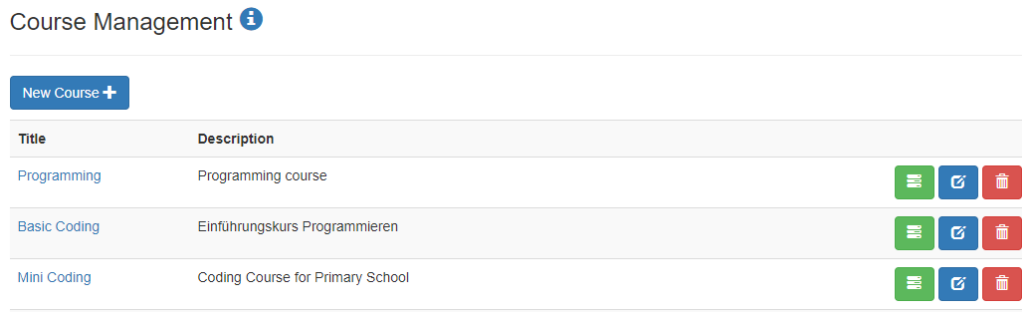


Figure 2.4: sCool Web application: overview of existing courses and new course create button (sCool version 3, year 2020; Steinmaurer et al., 2020; Steinmaurer et al., 2019; Mosquera et al., 2020;)

2.1.5 Overview

Each of the previously considered serious games has a particular focus on learning. Table 2.1 shows an overview of the previously discussed games and their features. In most games, the players attempt to solve tasks while controlling an avatar with their code. As players progress to higher levels, the task complexity also increases, and new concepts are introduced. Most of the games support novice players with no prior experience in coding. Most of the games provide an engaging environment to the players.

Table 2.1: Overview of serious games for computational skill learning

| Game | Platform | Language | Concepts | Dashboard |
|------------|----------------------|----------------------|---|-----------|
| CodeMonkey | Android, web browser | Python, CoffeeScript | Fundamentals, Control Flow, Algorithms, Objects | ✓ |
| Ozaria | web browser | Python, JavaScript | Fundamentals, Control Flow, Algorithms, Objects | ✓ |
| sCool | Android, Windows | Python | Fundamentals, Control Flow, Algorithms, Objects | ✓ |

2.2 Learning Analytics

The 1st International Conference on Learning Analytics and Knowledge defines Learning Analytics as:

“Learning analytics is the measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimizing learning and the environments in which it occurs” (Long and Siemens, 2011).

The focus of learning analytics is exclusively on the learning process. The learners and educators are the target beneficiaries. Learning analytics helps discover student’s interactions and performance, gain feedback, and improve course content to achieve effective learning. For educators, the availability of insight into learners’ performance can be a significant help in planning and improving course activities. For students, gaining information about their performance and progress compared to their peers or their own goals can be motivating. Learning analytics can help to improve the quality and value of the learning experience (Long and Siemens, 2011). Learning analytics is truly multidisciplinary with deeply connected roots to research areas such as statistics, text mining, big data, machine learning, human computer interaction, learning science and educational, cognitive and social psychology (Gasevic et al., 2016).

In the previous sections, some of the educational games for learning to code were discussed. Educational games generate humongous quantities of players game-related data. Tracking and storing this data is a straightforward task; however, making meaningful interpretations of the collected data is much more difficult (Seif El-Nasr and Canossa, 2013). Learning analytics tools and frameworks play a focal role in extracting meaningful information from the player’s collected game-related data. The collected data can be processed to extract meaningful information such as player’s progress and gain insights into their learning strategies. The extracted information can be made available to all interested stakeholders, where the stakeholders could be teachers, parents, students, and game developers. This information could

help teachers evaluate the progress of their students. The teachers could also help any student who may need assistance in understanding game courses. The information could also help game developers improve course content and identify various strategies used by the players.

2.2.1 Learning Analytics for Serious Games Architecture

Learning Analytics for Serious Game architecture comprises several modules. Individual modules interact with relevant stakeholders such as educators and students (Alonso-Fernandez et al., 2017; Freire et al., 2016). Figure 2.5⁴ shows an overview of serious game learning analytics architecture based on Alonso-Fernandez et al. (2017) and Freire et al. (2016).

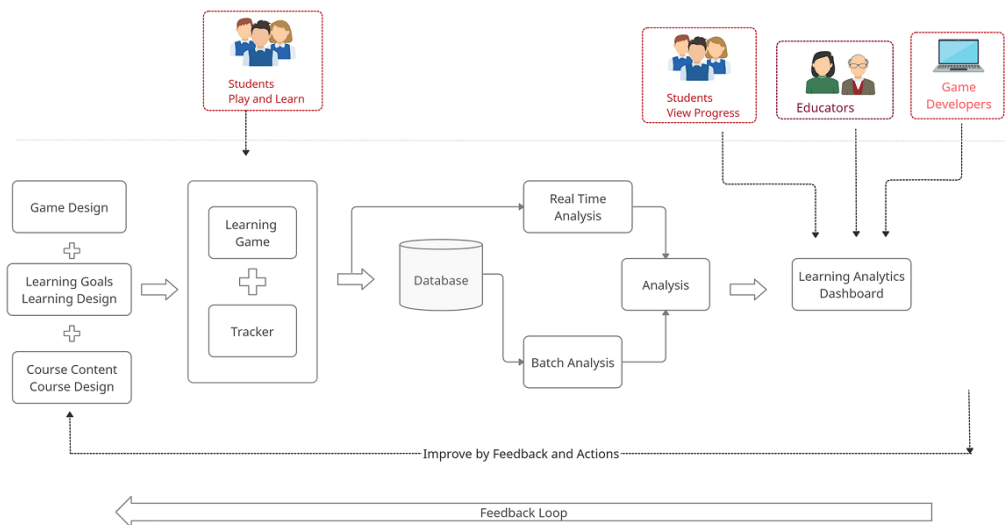


Figure 2.5: Overview of learning analytics for serious games architecture (after Alonso-Fernandez et al., 2017; Freire et al., 2016)

The steps in the process of learning analytics for serious games architecture can be described as:

⁴<https://app.creately.com/>

- The student’s data is collected, and the tracker (or a game data communication service) sends this data to be stored in a database.
- The relevant data in the database is processed and used for analysis. Specific live data can also be collected directly from the game.
- The data is processed to be used in the learning analytics dashboard and displayed in understandable formats to users using various visualization techniques.
- The educators can then gain insights from the learning analytics dashboard application and use these newfound insights to modify course content, assist students, or influence game design. This process runs as a feedback loop such that the course content and game keep evolving based on students’ needs.

2.2.2 Learning Analytics Information Model

Dashboards (Few, 2006) can provide a unique and powerful means to present important information to people. This information, when presented effectively, can provide them with an overview of data. Learning dashboards provide the present and historical information about a learner using various visualization techniques (Verbert et al., 2013).

Learning dashboards are also a specific subclass of personal informatics applications (Li et al., 2010). Adapted from the stage-based model of personal informatics (Li et al., 2010) and process model focused on learning analytics applications (Verbert et al., 2013), below stages of a model of how users process learning analytics application can be distinguished:

1. **Awareness:** Users view visuals of collected data as tabular overviews, graph plots, or other visualizations.
2. **Reflection:** Users reflect on the information by asking questions and assessing how useful and relevant the data is. This step involves looking at information

and exploring visualizations. Users reflect on information immediately (short-term information) or after several days (long-term information). For example, users looking at the course learning data every day or over a period, leading them to questions.

3. **Sense making:** Users answering the questions identified in the reflection process leads to the creation of new insights. Users explore and understand information. For example, users answering their course learning-related questions after further exploration.
4. **Action:** Users choose necessary actions based on newfound insights. For example, users taking necessary action such as adjusting course content, course topics, or their daily course schedule, and assisting a student with poor performance.

This framework presents a possible model of how users process information from learning analytics applications. It is a recursive flow, such that users follow the steps for improvements recursively. Figure 2.6⁵ shows an overview of the model discussed.

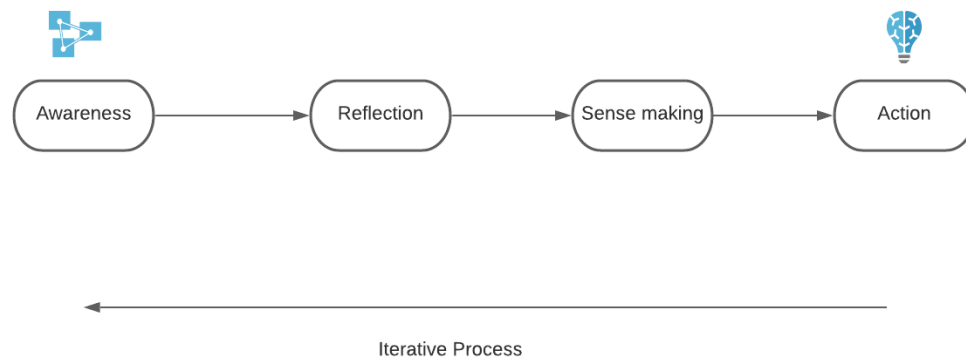


Figure 2.6: Model of users processing information from learning analytics applications (after Verbert et al., 2013; Li et al., 2010)

⁵<https://lucid.app/>

The learning analytics applications support teachers to gain insights into their respective courses (Step 1 - Awareness). They can then reflect on their course teaching and content (Step 2 - Reflection) and gain insights into their students who may require assistance, or regarding improvements and modifications in the course content (Stage 3 - Sensemaking). Following this, they can support these students respectively and update the course content (Stage 4 - Action) (Verbert et al., 2013). The course content could be too easy for students, or it could be too challenging. The educators gain insights from the learning analytics applications, and this process results in actions for their current course or future courses.

2.2.3 Code Analysis

The focus of this thesis is on learning analytics for educational games teaching programming or computational skills. Hence it is essential to analyze the programming solutions submitted by students. Analyzing programming solutions submitted by students could help educators understand how novice programmers learn and how to teach them best (Albluwi and Salter, 2020).

Many software engineering methods could be used to analyze code, which captures syntactic and semantic information embedded in the source code. Methods such as code clone detection (Kamiya et al., 2002; Sajnani et al., 2015), and bug localization (Zhou et al., 2012) utilize token sequences for representing programs (Zhang et al., 2019). These methods have been proposed to improve software development and maintenance.

The code clone detection tools (Kamiya et al., 2002; Sajnani et al., 2015) represents source code as a token sequence and this enables them to detect clones of code with a different line structures (Kamiya et al., 2002; Sajnani et al., 2015). The bug localization tool also performs lexical analysis of source code and generates a vector of lexical tokens (Zhou et al., 2012). However, these approaches may have a common problem of assuming the source code as being composed of natural language texts (Zhang et al., 2019). Even though the source codes are similar to plain texts, they contain richer and more explicit structural information. Panichella et al., 2013 also

discusses the findings that text in source code has different properties to natural language text and hence must be processed differently for better performance. Hence, recent works suggest that syntactic knowledge can obtain better representation than traditional token-based methods.

An Abstract Syntax Tree (AST) captures both the lexical and syntactical information of a source code (Zhang et al., 2019). AST serves as intermediate representation of program language and stores the syntax information of the source code (Baojiang Cui et al., 2010; S. Liu et al., 2020). The leaf nodes of AST usually represent identifiers and literals in the code. The non-leaf nodes can represent some syntactic structures.

Listing 2.1 shows a simple python program with an assignment statement where a constant is assigned to a variable *a*. Figure 2.7 visualizes AST generated of the Python program. The *module* is the base class, and the *assign* is the child node with other child nodes of *variable* and *constant*. AST could also provide information about the code constructs used in the program. Such as a variable, a constant, and an assignment was used in this program.

```
1 a = 2
```

Listing 2.1: A simple python program with an assignment statement

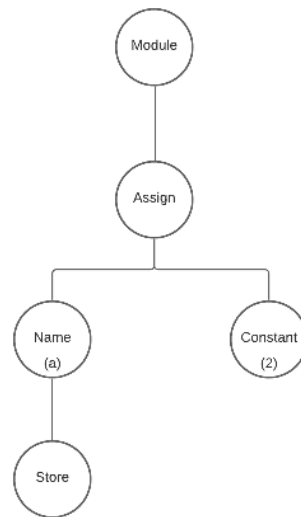


Figure 2.7: Structure of AST tree generated of code statement from listing 2.1

As an AST is a high abstraction of source code, it could help analyze submitted code solutions by students. It could provide an overview of the various code concepts learned and eventually used by students in solving coding tasks as part of the courses teaching programming or computational skills.

2.2.4 Learning Analytics Tools and Dashboards

In this section, an analysis of some of the learning analytics tools and their supported features are presented. Learning analytics tools and dashboards are designed for stakeholders such as educators, learners, parents, and game developers. Various learning analytics tools track information such as the track of time spent, social interactions, activities students participated in, and learning content usage. The game-specific tools provide learning analytics about players of their respective games. The learning analytics information is being made available to stakeholders using web applications in many of the tools.

2.2.4.1 CodeMonkey

CodeMonkey⁶ is an apps and web-based educational game where kids learn to code (CodeMonkey, 2021).

CodeMonkey provides a teacher's dashboard where teacher's can keep track of student's progress and see the actual code student's submitted. Teacher's also have access to the task solutions so they can compare student's solutions. It supports an entire classroom management system with features such as creating a curriculum and automatic grading. The student's can be assigned to a classroom and managed by teachers. Teacher's can also add or remove additional teacher's to their courses.

Figure 2.8 shows CodeMonkey's teacher's dashboard. The teachers can see the progress of each of the participant students in their course. The teachers can see completed exercises by the students. Teachers can also see a list of all students who have currently joined their course and remove unwanted students. Teachers can also export results and progress, or receive a more detailed analysis. Figure 2.9 depicts CodeMonkey's dashboard for teachers showing the overall grades of students. In the grades section, the teachers can see each participant's grades and a class overview with information such as the number of students in the class who have completed the course successfully. Figure 2.10 shows students' proficiency in computer science topics based on completed course and course exercises. The teachers can see all the computer science concepts such as for loops, functions, simple loops, and objects and students proficiency in each of them respectively.

⁶<https://app.codemonkey.com/>

CHAPTER 2. BACKGROUND AND RELATED WORK

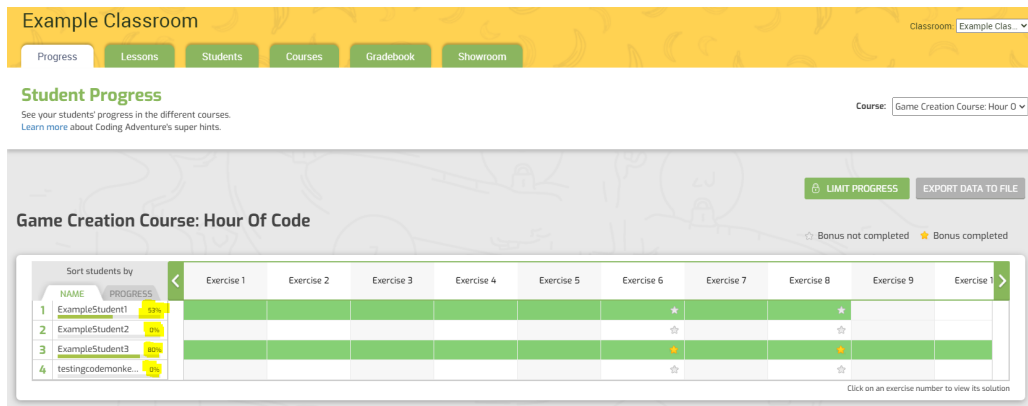


Figure 2.8: CodeMonkey dashboard for teachers showing students progress in the course

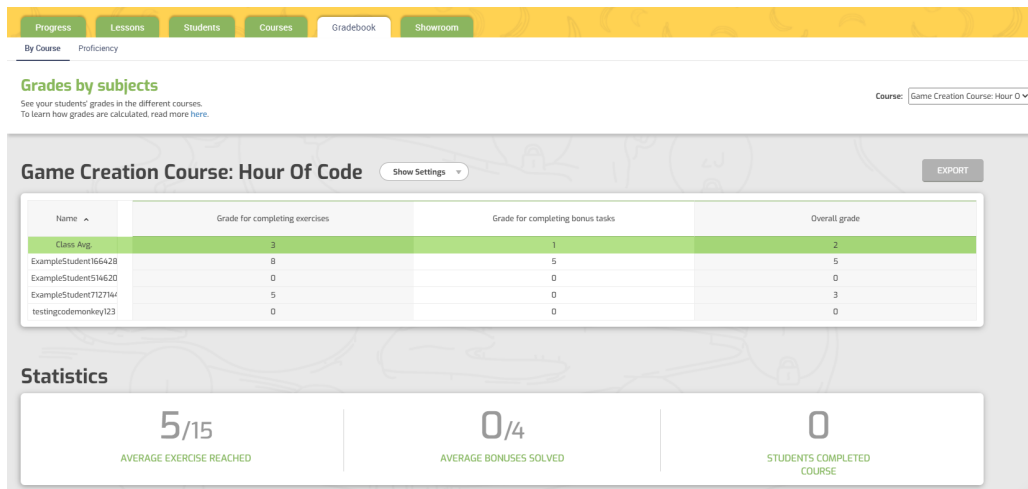


Figure 2.9: CodeMonkey dashboard for teachers showing students overall grades

CHAPTER 2. BACKGROUND AND RELATED WORK

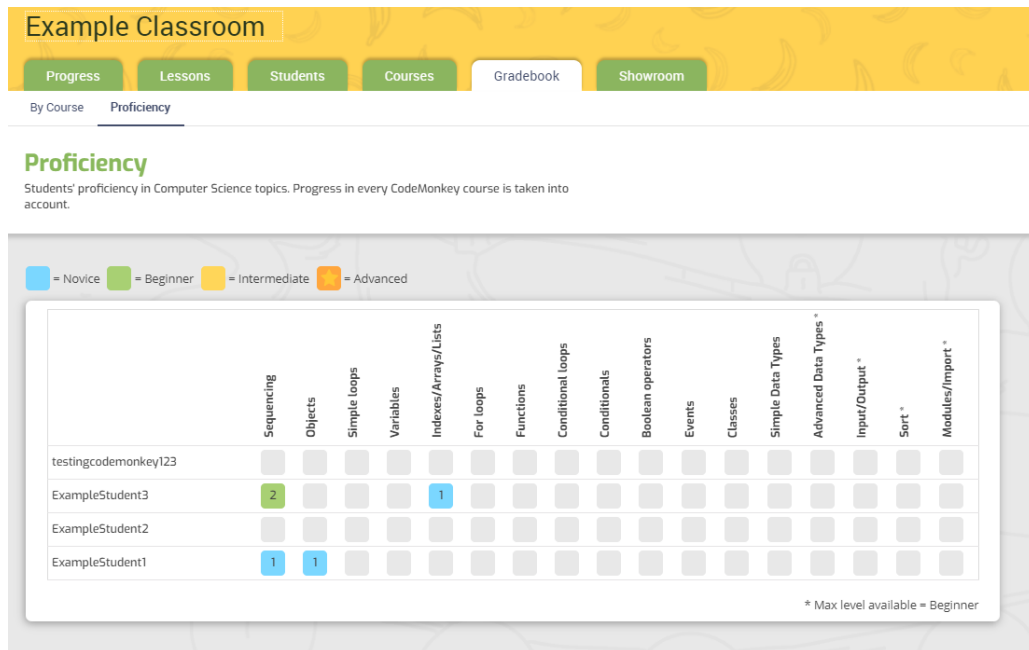


Figure 2.10: CodeMonkey dashboard for teachers showing students proficiency in computer science topics

CodeMonkey also provides a dashboard for students where students can see available courses and progress in the respective courses. Figure 2.11 shows CodeMonkey’s student’s dashboard.

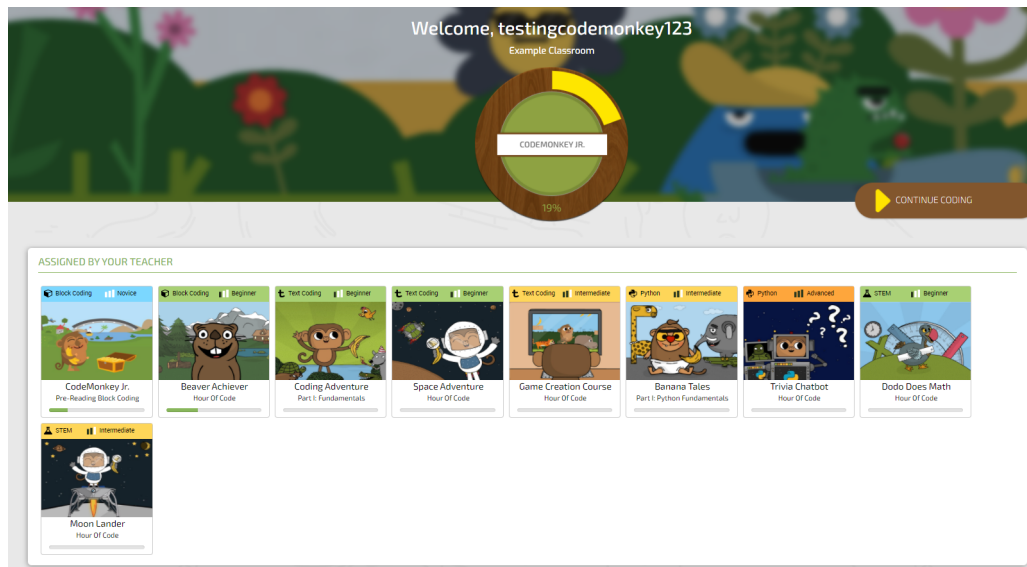


Figure 2.11: CodeMonkey dashboard for students showing progress in current course and available courses

2.2.4.2 Ozaria

Ozaria⁷ is a web-based serious game for learning coding skills, as discussed previously. Ozaria provides a comprehensive and intuitive dashboard for teachers, as well as students. The teachers have access to student lesson plans, pacing guides, and progress tracking. Through progress tracking, teachers can quickly identify students who are struggling and see their submitted solution. Ozaria also provides a student dashboard where students can see their classes and progress in each class.

Figure 2.12 shows Ozaria dashboard for teachers showing a class view with students and their progress. The teachers see all the levels completed by the students and the currently in-progress and assigned levels. The teachers can also add or remove students from their class. The teachers can also sort students by last name, first name, progress high to low, and progress low to high. The teachers also have access to a Ozaria course curriculum guide, with information about all the course levels and the coding concepts covered in the course curriculum. Figure 2.13 shows

⁷<https://www.ozaria.com/>

CHAPTER 2. BACKGROUND AND RELATED WORK

the feature options where teachers can view code submitted by students. However, during testing, some of the information was not accessible or was static content. The tool may be a work in progress, and the features may be available in future releases.

In summary, some of the main information which Ozaria teacher's dashboard provides are as:

- List of all classes
- Students lesson plans
- Progress tracking: educators can see individual progress made by students. Educators can identify struggling students with less course progress and in need of assistance for improvements with course progress.
- Modify courses
- Submitted code solutions by students

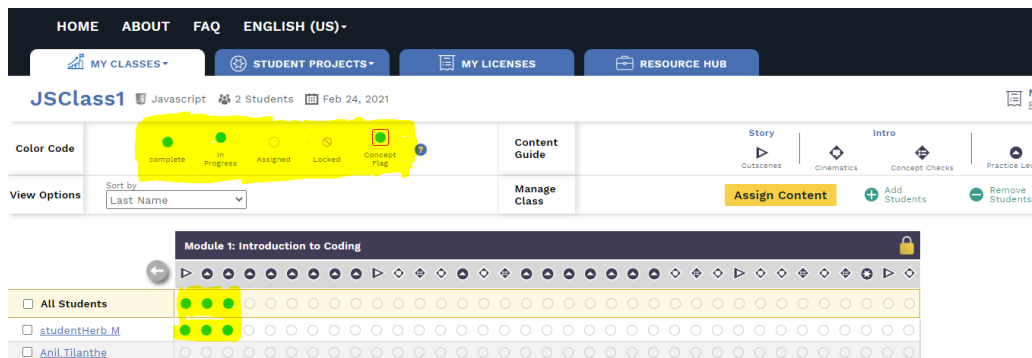
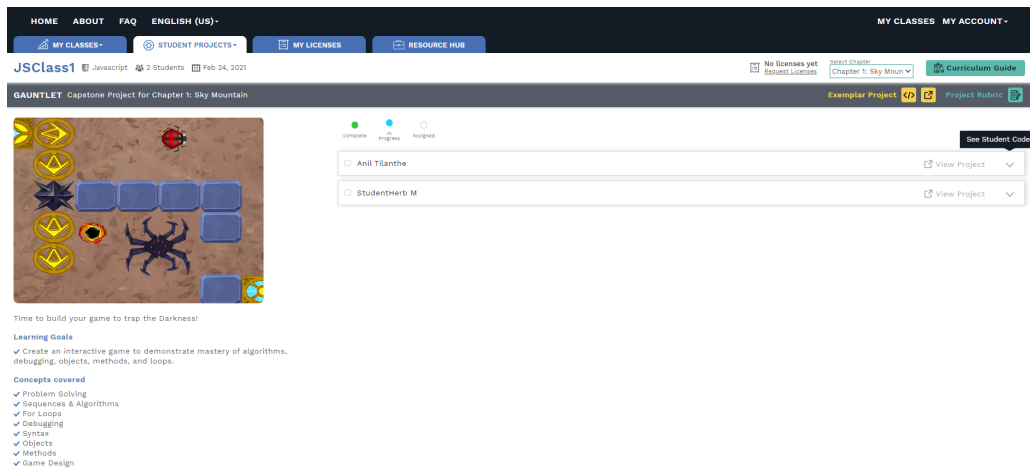


Figure 2.12: Ozaria dashboard for teachers class view depicting students progress in the course and other features

CHAPTER 2. BACKGROUND AND RELATED WORK



The screenshot displays the Ozaria dashboard for a teacher's class view. The top navigation bar includes links for HOME, ABOUT, FAQ, ENGLISH (US), MY CLASSES, STUDENT PROJECTS, MY LICENSES, and RESOURCE HUB. The main content area is divided into two sections. On the left, there is a game preview titled "GAUNTLET: Capstone Project for Chapter 1: Sky Mountain" with a "See Student Code" button. Below the game preview, there are "Learning Goals" and "Concepts covered" sections. The "Learning Goals" section includes a checklist: "Create an interactive game to demonstrate mastery of algorithms, debugging, objects, methods, and loops." The "Concepts covered" section includes a checklist: "Problem Solving", "Sequences & Algorithms", "For Loops", "Debugging", "Syntax", "Objects", "Methods", and "Game Design". On the right, there is a list of student submissions with two entries: "Anil Tilanthe" and "StudentHerb M", each with a "View Project" link.

Figure 2.13: Ozaria dashboard for teachers class view depicting code submissions of students

Ozaria also provides a dashboard for students to see the classes they have joined and progress in respective courses. Figure 2.14 shows Ozaria student's dashboard.

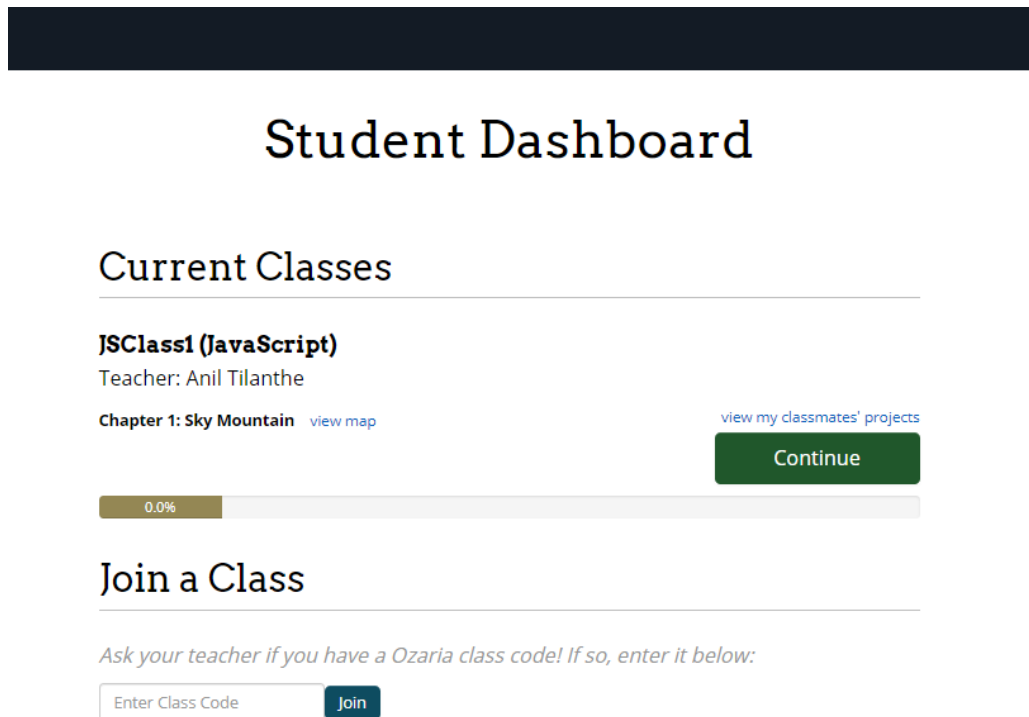


Figure 2.14: Ozaria dashboard for students depicting joined classes and progress in current class

2.2.4.3 sCool

sCool is a serious game-based educational tool for learning coding or computational skills (Steinmaurer et al., 2020; A. Kojic, 2017; Steinmaurer, 2019; Mosquera et al., 2020; Steinmaurer et al., 2019). The gaming environment consists of two components: a) sCool mobile game and b) web application.

The sCool web application supports the adaptive learning content. Educators can create new learning content for their courses as well as modify and update them when required. Educators can define learning content for both concept-learning and practical missions. Educators also have access to analysis of participating students where educators can see the course progress of the students (Steinmaurer, 2019).

sCool does not provide a separate dashboard for student's, however, progress can

be seen by students within the game itself.

2.2.5 Overview

A simplified architecture of a serious game learning analytics tool consists of the serious game and a tracker. The tracker sends information such as player's game-based data to be stored in the database. The player's game-based data is then processed for analysis and displayed in understandable formats using various visualization and exploration techniques. Educators gain insights from the learning analytics tool and could use the knowledge gained to improve course content or assist students in need.

Users process information from learning analytics tools such that they gain awareness by viewing visuals of collected data. They reflect on the information by asking questions and assessing how valuable and relevant the data is. Users then start answering the question and making sense of the data, which leads to newfound insights. Users then choose suitable actions based on newfound insights such as updating course content, assisting students in need, or updating schedules.

Various learning analytics tools have features and capabilities to explore and gain insights from student's data. Educators can view student's grades, student's progress, and code submitted, among other features. Most of the tools are web-based tools for teachers.

Table 2.2 provides an overview of learning analytics tools and dashboards for educational games for learning coding or computational skills. Each of them provides various features and varying degrees of information to the users. The tools are listed with their features and availability for teachers and students. The teacher's dashboard provides information about the student participants and their progress in the courses. Teachers have access to various features such as viewing students' submitted code, students' grades, student's scores, and adding or removing students from the courses. Some of the learning analytics tools also provide code analysis of the submitted code and concepts used or learned by students. All the student's dashboards provide information to students about their course progress. sCool pro-

vides an overview of course progress within the mobile game itself, and there is no separate dashboard available. There is a lack of features of sCool’s learning analytics dashboard. Additionally, it could be concluded that sCool’s learning analytics dashboard could be further developed, providing more information to the stakeholders. Providing additional learning analytics information to sCool’s stakeholders, such as educators and game developers, could enable them to improve the game content and identify students in need of assistance.

Table 2.2: Overview of learning analytics tools and dashboards for educational games

| Name | Teachers | Students | Features (for teachers) | Features (for Student) |
|------------|----------|----------|---|--|
| CodeMonkey | ✓ | ✓ | Progress of students. Code submitted by students. Students Grades. Code submitted by students. Analysis of submitted code. Code concepts used. | Available courses. Progress in each course. |
| Ozaria | ✓ | ✓ | Progress of students. Code submitted by students. Student’s lesson plans. Code submitted by students. | Current classes. Progress in each class. |
| sCool | ✓ | ✓ | Progress of students. List of students. | In game course progress |

2.3 Data Visualization

Pictures have been used for communication since before written languages. Pictures depicting stories from neolithic humans have been found in caves and other places. Figure 2.15 shows an rock painting from the Cave of Beasts (Gulf Kebir, Libyan Desert) which is Estimated 7000 BP old. Pictures can also be processed more quickly compared to a page of words (Ward et al., 2010).

“Visualization is the communication of information using graphical rep-

resentations” (Ward et al., 2010).

Visualization helps us gain insights into data through interactive graphics. Insights into a piece of complex information, data, or process can be conveyed using one or several graphics (Telea, 2014). Data visualization breaks down complex data into something which human minds could process easily. Data visualization is an extensive field and includes mathematics, computer science, cognitive and perception science, and engineering (Telea, 2014). Visual analytics combines visualization with analytics, and has become a key area of interest in the field of data visualization (J. Liu et al., 2018).



Figure 2.15: Rock paintings from the Cave of Beasts (Gifl Kebir, Libyan Desert) Estimated 7000 BP (Schmillen, 2014)

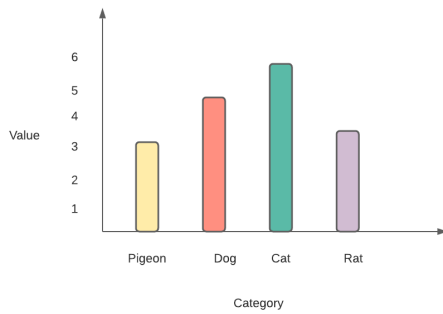
2.3.1 Visualization techniques

Some of the common visualization techniques are as (Sadiku et al., 2016; Kosara, 2016; Jarrell, 1994):

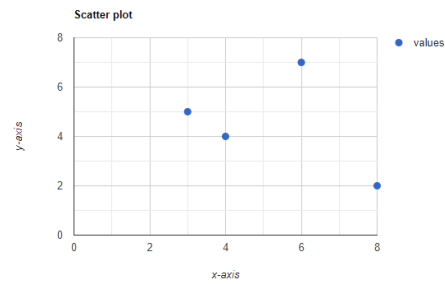
- **Bar chart:** Bar charts present categorical data with rectangular or cylindrical bars with heights or lengths proportional to the values they represent (Bar chart, 2021). The graphs can be plotted vertically or horizontally. The bar chart shows a comparison for discrete categories where one axis is the category and another axis the value. Figure 2.16a shows a simple bar chart depicting number of animals in a household.
- **Scatter plot:** Scatter plot display values for typically two variables of a data set (Jarrell, 1994; Scatter plot, 2021). The points on the scatter plot can be coded with colors, size, or shape to represent an additional third variable. Each of the points has variables determining position on the horizontal and vertical axis. Figure 2.16b shows a simple scatter plot.
- **Pie chart:** A pie chart is a statistical chart displaying slices with size proportional to the values represented (Siirtola, 2014). Figure 2.16c shows a simple pie chart.
- **Line chart:** Line graphs display information as a series of data points connected by line segments. It is similar to the scatter plot, except that the data points are ordered and connected by line segments. It is used to visualize changes or a trend over a time period. Figure 2.16d shows a simple line chart.
- **Tables:** Tables show values of a data set in rows and columns. The table could be either column-based or row-based. In a column-based table, the column header or the first value of a column contains the feature label, followed by the data set's feature values, for example, columns of name, height, and weight of students of a school. Tables could also have multiple dimensions, such as representing a 2-dimension matrix as a table.
- **Maps:** Maps could be used to represent spatial relationships between data by representing the data points on a geographical map. Even the size of data points could also contain additional information. For example, a map chart representing the spread of Covid-19 virus around the world. The size of data points could represent the number of cases in past seven days.

- **Gantt Chart:** Gantt Chart shows the schedule of a project or its timeline, and it was named after its inventor, Henry Gantt (Gantt, 1910). It is a type of bar chart displaying project schedules (Klein, 2001).

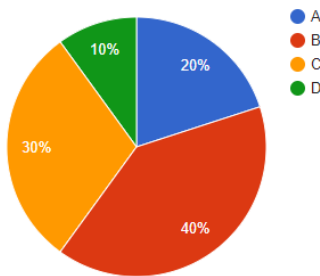
Data to visualize can be lower-dimensional data or higher-dimensional data (Keim, 2002). Many complex visualization techniques could be used to display complex data as higher-dimensional data may require more complex visualization techniques to depict the data. The focus should be on lower-dimensional plots such as one-dimensional, two-dimensional, or three-dimensional plots to keep the visualizations easier to understand for the stakeholders of our educational games (Keim, 2002). The visualization tools and frameworks enable users to create visualization charts with simple configurations.



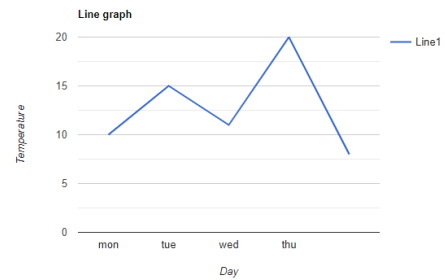
(a) Bar chart



(b) Scatter plot



(c) Pie chart



(d) Line chart

Figure 2.16: Data visualization techniques

2.3.2 Visualization Tools and Frameworks

Modern visualization tools have integrated useful functions such as data pre-processing and visualization into libraries, plugins, or configurable settings (J. Liu et al., 2018). The analytics dashboards give the ability to create highly interactive dashboards and content with visual exploration (Richardson et al., 2018). Interactive visual exploration enables data exploration with interactive visualization options such as scatter plots, heat and tree maps, time series plots, and geographic maps (Richardson et al., 2018). Modern visualization tools not only convert raw data into visual graphics but also make the data easily human-understandable. Modern visualization tools and frameworks provide an essential mechanism to transform complex information from learning analytics tools into easy-to-understand graphs and visuals for essential stakeholders such as teachers, students, and parents.

Many of the visualization tools and frameworks are interactive. Modern visualization tools and frameworks enable users to interact with the visualizations in ways such as hover over the information, zoom in and out of the visualizations and select a subset of the data (Ali et al., 2016).

Below some of the modern visualization tools and frameworks are discussed. A broad classification based on the requirement of programming knowledge is considered as: a) Tools and Frameworks with No Programming and b) Tools and Frameworks with Programming (J. Liu et al., 2018). These are presented in the following subsections.

Tools and Frameworks with no Programming

Visualization tools without programming provide easy to use interface to users. They enable users to create standard charts or graphical visualizations. Users can import data to these tools. Below some of the visualization tools and frameworks with no programming, and their features are discussed.

2.3.2.1 Tableau

Tableau⁸ is a visual analytics platform for business intelligence and analytics. It was founded in 2003 due to a computer science project at Stanford to make data more accessible using visualizations (tableau, 2021b). It supports various data import options such as .csv, .txt, .xls files, and importing from external servers, such as salesforce data and MySQL. Users can easily establish a secure connection to a data source from Tableau. The columns from data source are shown as fields. Users can then use drag and drop to select these fields and choose the visualization type to generate the final visualization. Figure 2.17 shows an example Tableau COVID-19 global tracker dashboard showing cumulative cases worldwide.

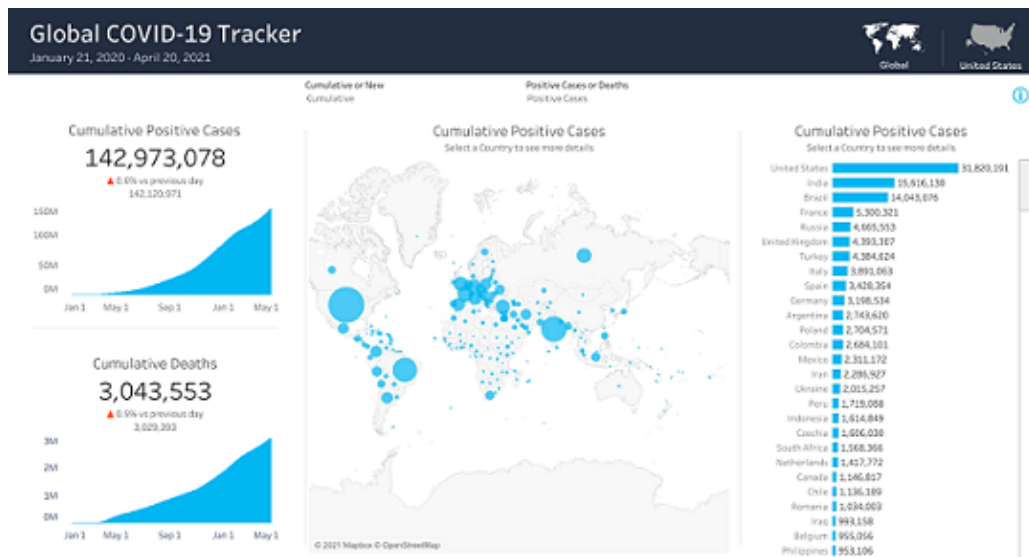


Figure 2.17: Tableau Covid-19 global tracker dashboard (tableau, 2021a)

2.3.2.2 Infogram

Infogram⁹ is a web-based application for data visualization. Users can register on the web platform. Users can then upload data and select the type of chart to create

⁸<https://www.tableau.com/>

⁹<https://infogram.com/>

quick data visualization (Infogram, 2021a). Infogram supports reusable templates, which results in reduced configuration time. Figure 2.18 shows an example Infogram template dashboard from example templates of the application.

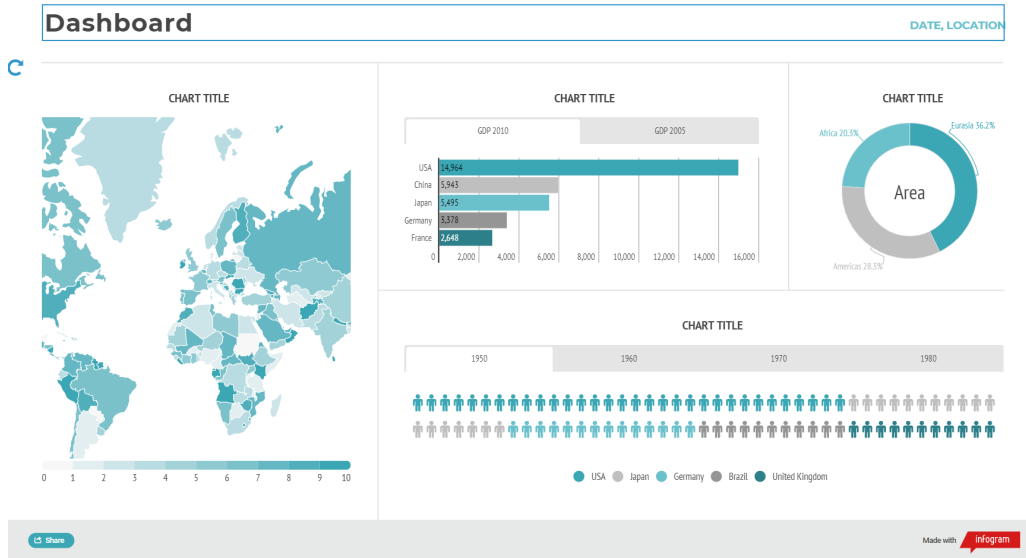


Figure 2.18: Infogram template of a sample dashboard (Infogram, 2021b)

2.3.2.3 RAW Graphs

RAW Graphs¹⁰ is an open-source data visualization framework. RAW Graph is *"built with the goal of making visual representation of complex data easy for everyone"* (RAWGraphs, 2021). It can be used directly without registration. It supports various data formats such as .csv, .xls files. It processes data in the web browser itself. It allows users to choose chart type and map dimensions by dragging features into variable attributes. Generated charts can also be exported as an image, a vector (SVG), or embed into web pages using generated code. Figure 2.19 shows an example RAW graph visualization of movies dataset where the x-axis is production budget, and the y-axis is box office performance.

¹⁰<https://rawgraphs.io/>

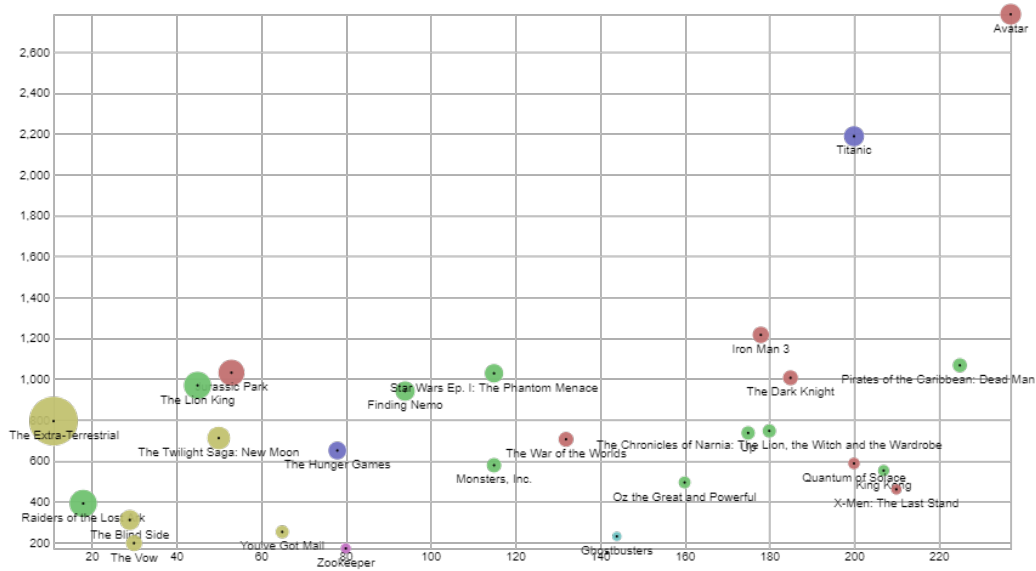


Figure 2.19: RAW Graph movie visualization of production budget vs box office (graph, 2021)

Tools and Frameworks with Programming

Visualization tools without programming are easy to use for any user. Using customizable properties, they enable all users to create standard charts or graph visualization. However, they may have limitations such as limited customization and integration possibilities. Some visualization tools have Application Programming Interface (API) to enable developers to integrate them and create custom-enhanced visualizations based on requirements. Below some of the visualization frameworks with programming which provides high customization are discussed.

2.3.2.4 D3.js

D3.js¹¹ is an open-source JavaScript graphics library using Scalable Vector Graphics (SVG), HyperText Markup Language (HTML5), and Cascading Style Sheets

¹¹<https://d3js.org/>

(CSS) standards for producing dynamic, interactive data visualizations for web browsers (Fan Bao and Jia Chen, 2014; Comparison of JavaScript Charting Libraries, 2021). D3.js has plenty of documentation support, tutorials, and examples on its website for learners. Figure 2.20 shows an example map of the d3.js library depicting county population in the USA from documentations⁽¹²⁾.

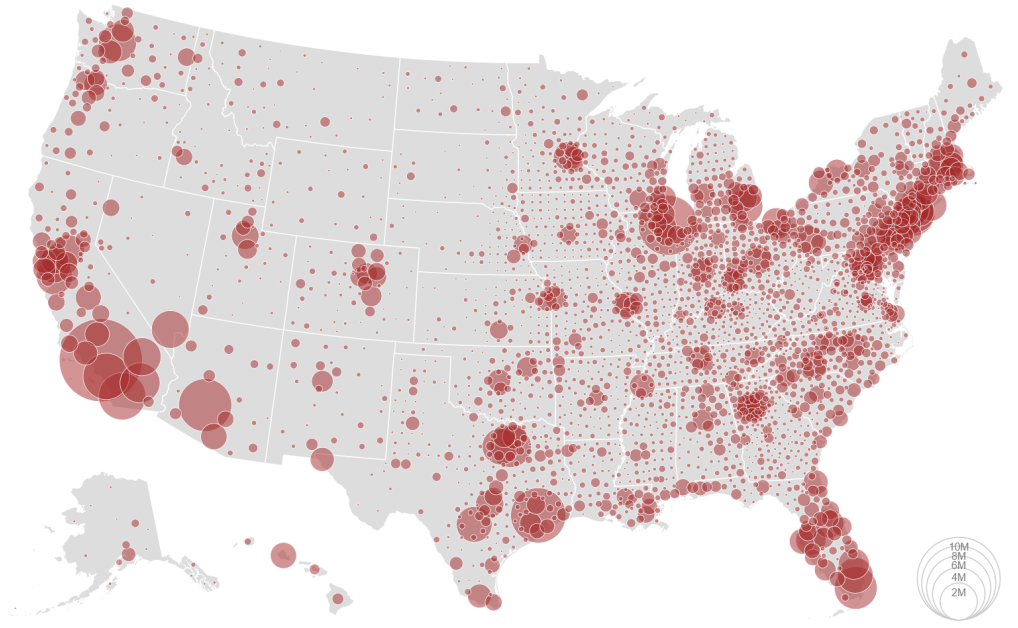


Figure 2.20: D3.js example bubble map : county population in USA (Bostock, 2014)

2.3.2.5 Chart.js

Chart.js¹³ is an open-source JavaScript graphics library. It uses Canvas on HTML5 for great rendering performance across all modern browsers. It also supports responsiveness on windows resizes for perfect scale granularity. It supports different chart types, which can be used to visualize data in different ways and each of them is animated and customizable. Figure 2.21 shows an example chart type of

¹²<https://bost.ocks.org/mike/bubble-map/>

¹³<https://chartjs.org/>

Chart.js library depicting sample bubble chart from documentations⁽¹⁴⁾.

Bubble

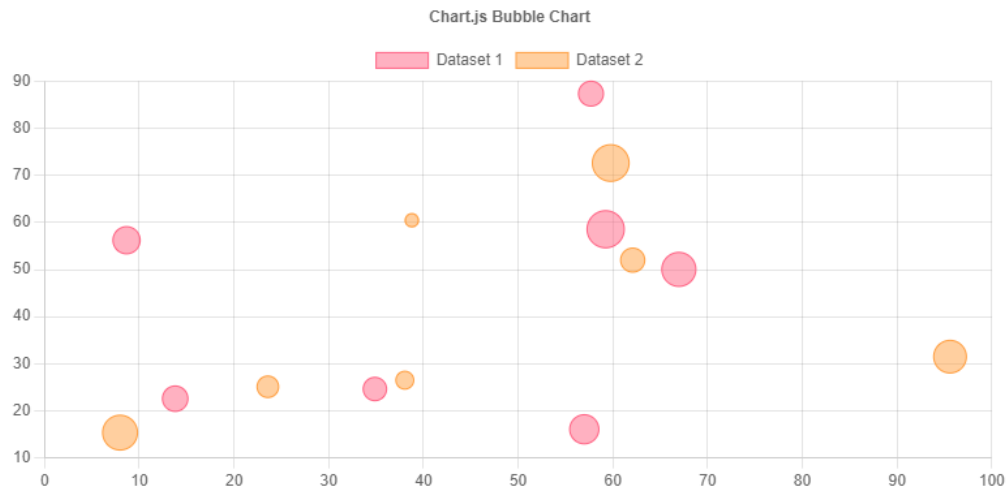


Figure 2.21: Chart.js example bubble chart (Bostock, 2021)

2.3.2.6 Plotly

Plotly¹⁵ has open-source graphing libraries in Python, JavaScript, and R to create dynamic, interactive charts and visuals. It uses plotly.js JavaScript graphics library internally, which is based on D3.js and WebGL (Mease, 2018). Using Python and R languages enables users to process complex data and models and then use the Plotly libraries to create interactive visuals. Plotly can also integrate with the Dash framework for building web-based analytic applications. Figure 2.22 shows an example of Plotly mixed subplots from documentations⁽¹⁶⁾.

¹⁴<https://www.chartjs.org/docs/latest/samples/other-charts/bubble.html>

¹⁵<https://plotly.com/>

¹⁶<https://plotly.com/python/mixed-subplots/>

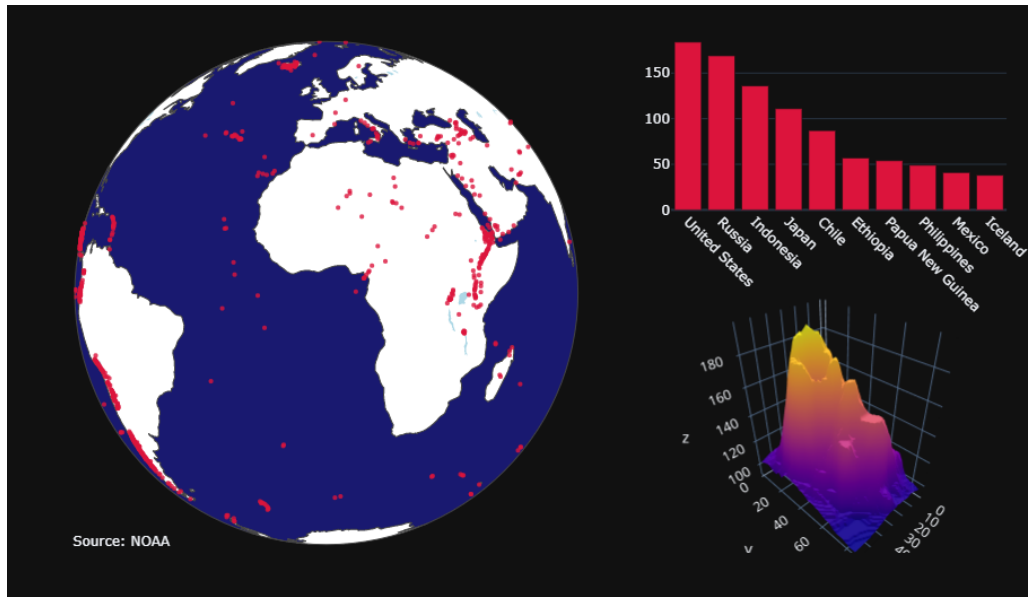


Figure 2.22: Plotly example mixed subplots from documentations (Plotly, 2021)

2.3.3 Overview

Data visualization tools and frameworks could convey knowledge gained from learning analytics to essential stakeholders such as teachers, students, and parents in easy to interpret ways. Graphs such as time series plots could even show student's interaction timelines with the games. The time series plots could also provide information about the student's strategies by describing their route undertaken to complete the course.

In Table ?? lists data visualization tools and frameworks and their features respectively. Each of data visualization tools and framework provides various features and varying degrees of usability. Some of them are open source, and others are commercially licensed. Each of the visualization tools and frameworks has its merits and demerits, and a suitable one would be the one that matches the use case and available resources of money, knowledge, and time.

2.4 Summary

Various educational games for learning to code provide immersive stories and content to keep students motivated and engaged while learning programming concepts. The educational games are available for various platforms such as Windows or mobile devices. The educational games also generate and collect a massive amount of learner data. This data can be used to gain further insights and improve the learning experience of learners. With game metrics, among other information, developers and educators can gain insights into their users and improve the learning experience accordingly. Various educational games offer learning analytics dashboards for stakeholders such as teachers, parents, and students.

The chapter also discussed learning analytics tools for educational games for learning coding or computational skills, and their features. Some of the learning analytics tools provide information such as code submitted by students and analysis of the submitted code to the teachers in a web application. The code learning game sCool could provide additional learning analytics features to its stakeholders such as educators, and game developers. The additional learning analytics features of sCool could enable the stakeholders to improve game content and help students in need.

The chapter further discussed the features of various data visualization frameworks and tools. All of the tools have merits and demerits. Some of the visualization tools and frameworks have benefits such as being open source and free to use, and other tools offer ease of use and development for users. It could be conclude that the best data visualization tool or framework would be the one that matches our requirements and available resources.

The next chapter describes the sCool learning game system in detail. Various components of the sCool learning game system are described further to surmise the possibilities of additional learning analytics capabilities for the various stakeholders of the sCool learning environment. Such capabilities could assist various stakeholders of the sCool learning system to improve game content and gain insights into students' performance.

Chapter 3

sCool

In this chapter, the description of the mobile learning game sCool and its environment is given. The chapter describes the game design and system architecture of sCool. The game's main components comprising of the sCool mobile game and web application are further described.

3.1 Game Design

The mobile learning game sCool was initially developed in 2017 in cooperation with Graz University of Technology and Westminster University (A. Kojic, 2017; Steinmaurer, 2019; Steinmaurer et al., 2020; Mosquera et al., 2020; Steinmaurer et al., 2019). It is a game-based learning tool for computational skills. The game's engaging story seamlessly links various game modes within the game.

The game follows the theme of space and exploration of a foreign planet. The players are part of a space shuttle that crashed on a foreign planet (A. Kojic, 2017; Steinmaurer, 2019). Some of the space shuttle parts are lost. The players play a crew member's role and need to repair the shuttle by exploring the planet and collecting pieces of information. Each piece of information represents a learning content that helps players understand a particular programming concept. The information about programming concepts is followed by a question regarding the learned concept.

After successfully passing concept-learning missions and understanding the concept, they apply their knowledge in the practical missions of controlling a robot using the programming language Python and understand computational thinking skills while collecting lost parts of their space shuttle (Steinmaurer, 2019).

3.2 System Architecture

sCool gaming environment consists of two components: a) sCool mobile game and b) web application. The sCool mobile game consists of two game modes: i) concept-learning mode and ii) practical mode.

Figure 3.1 visualizes the architecture of sCool environment. The web application plays a vital role since it provides the game with learning content. Educators can create learning content for their courses using the web application. The Player's game-related data collected by the mobile game is transferred to the server and stored in the sCool database. All communication between the web application and mobile game is done using Representational State Transfer (REST) APIs, and data is transmitted in JavaScript Object Notation (JSON) format. After each level, all game data is synchronized with the server (Steinmaurer, 2019).

Learning content in sCool is highly adaptive as educators can create and modify individual courses for their classes. The web platform displays courses in a hierarchical skill tree. Educators can define content for both concept-learning and practical parts, which is then accessible to their students in the sCool game. With this approach, educators can provide an individual learning experience and update course content based on feedback loops (Steinmaurer, 2019).

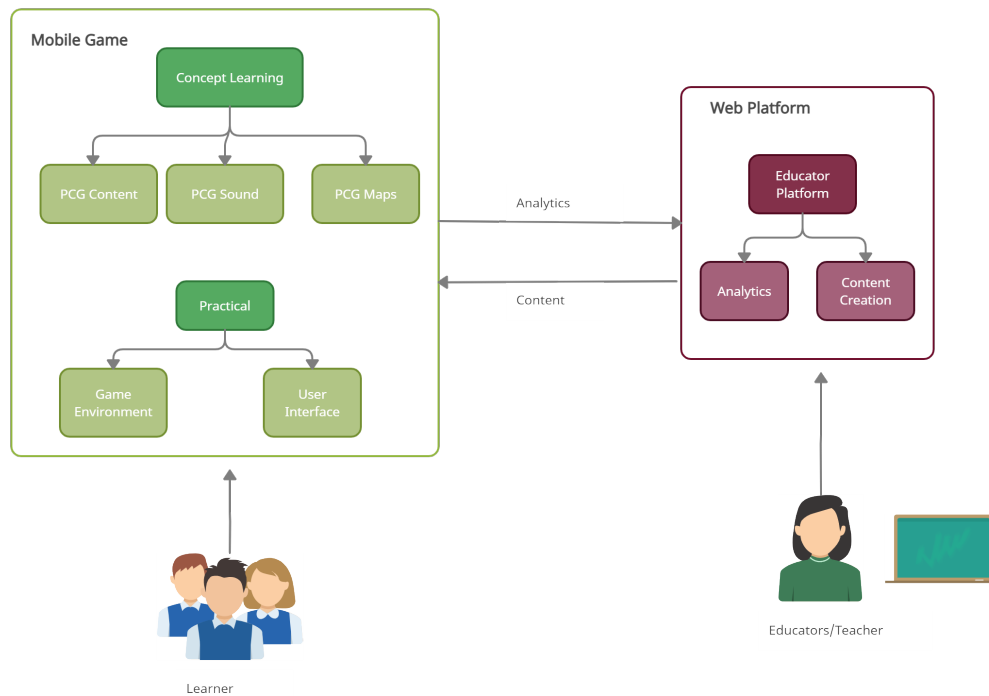


Figure 3.1: sCool - System Architecture (adapted after Steinmaurer et al., 2020; Steinmaurer, 2019)

3.3 Mobile Game

The sCool mobile game is currently available for Microsoft Windows and Android devices. The game combines two different modes with different objectives. The first mode is a concept-learning mode that introduces players to new programming concepts. In a procedurally generated game world, players have to find disks representing concepts while defeating alien enemies who are guarding these disks. After collecting disks, the concepts are presented in textual form followed by a related question. The mobile game also provides an in-game overview of course progress to students. The in-game overview of course progress helps the students keep track of their progress for course completion and make various strategic decisions. Figure 3.2 shows the concept of learning game mode where players have to find and collect disks guarded

by enemies. Figure 3.3 shows the concept learning game mode where players retrieve the disks, and then they are introduced to new programming concepts. Figure 3.4 shows the concept learning game mode where related questions follow an introduction to a new programming concept.



Figure 3.2: sCool Concept learning. Players find disks guarded by enemies. (sCool version 3, year 2020.)



Figure 3.3: sCool Concept learning. When Players retrieve disks they are introduced to new programming concepts. (sCool version 3, year 2020.)

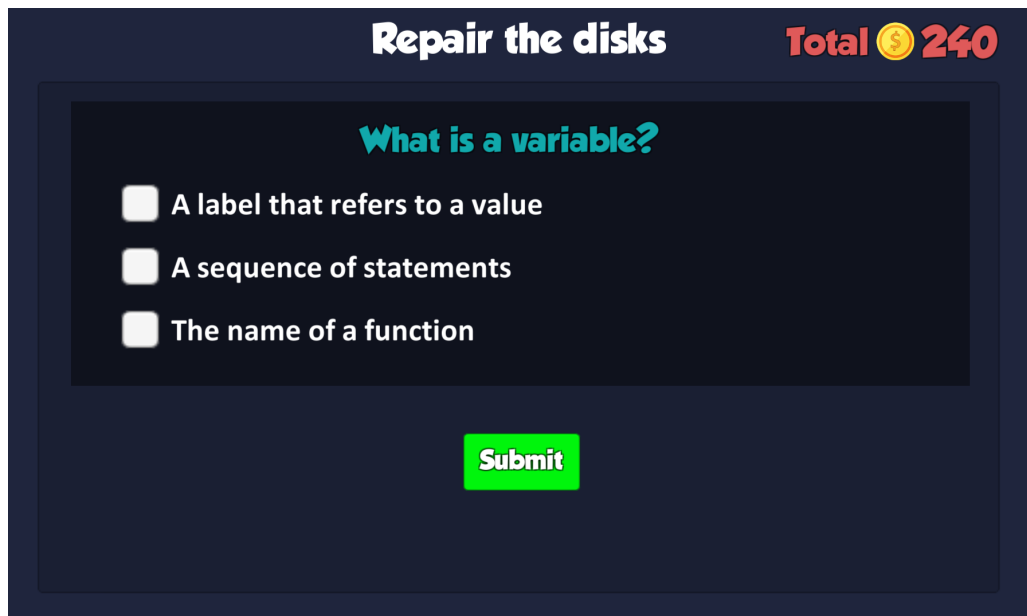


Figure 3.4: sCool Concept learning. Introduction to new programming concepts is followed by related questions. (sCool version 3, year 2020.)

The second mode is the practical missions, where players complete tasks with a practical application of previously learned programming concepts. The second part playground is a 2-dimensional grid where players control and move a robot using code. The goal in each level is to reach a disk while solving various tasks. The robot can be controlled by specific instructions that represent commands in the Python programming language. There are draggable code blocks in the user interface that get converted to Python commands that the robot can execute. There is also a virtual keyboard to edit or write code (Steinmaurer et al., 2019). Figure 3.5 shows the practical learning game mode where players utilize the previously learned concepts and control a robot using programming instructions to collect disks for their space shuttle.

Various programming concepts (Mitchell and Apt, 2003; Strachey, 2000) can be taught using tasks in sCool game for a basic course under K-12 education (Steinmaurer, 2019). For example, conditions, arrays, variables, loops, constants, classes, and objects can be taught using tasks in sCool mobile game (Steinmaurer, 2019).

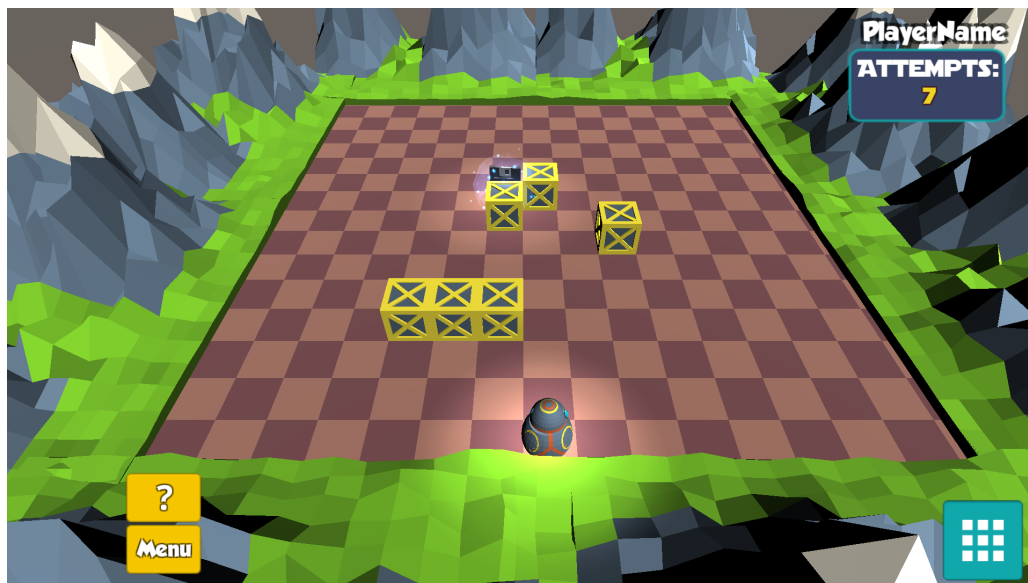


Figure 3.5: sCool Practical learning. Robot in a 2-dimensional grid with a disk to reach. (sCool version 3, year 2020.)



Figure 3.6: sCool Practical learning. Code interface with code blocks and other options. (sCool version 3, year 2020.)

3.4 Web Platform

The web application is a .NET based platform for educators (A. Kojic et al., 2018; Steinmaurer, 2019). It consists of the following components:

- Content creation for adaptive learning
- Adaptive learning mechanism with editable and adaptable course content
- Web APIs for providing learning content and receiving analytics data from the mobile game
- Overview of course students

Educators can use the web platform to create new courses, or to edit and update content of their existing courses. Courses are represented as in a hierarchical skill-tree. Figure 3.7 visualizes the hierarchical tree. Skills are the superior elements for

all tasks in concept learning mode and practical mode. Each task is a subset of a skill. Various concepts can be mapped to a skill tree.

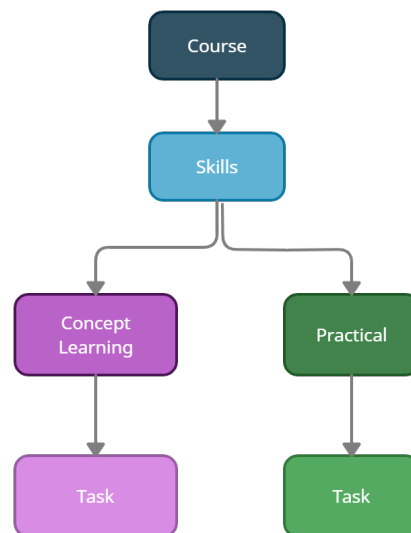


Figure 3.7: sCool Hierarchical Course Tree (redrawn after Steinmaurer, 2019; Steinmaurer et al., 2020)

The web application lets educators declare the content and the degree of difficulty as well. This information is then used in the game to generate maps based on provided course parameters. The web platform also distinguishes between the concept learning mode and the practical mode. In the concept learning mode, educators can define the learning content and related questions. The choice of difficulty level for the exploration mode is also available on the web platform. This difficulty level value is then used to generate the map accordingly. For the practical tasks, educators can define the task goal and output solution. This output solution represents the expected output of the Python code in order to complete the task. It is also possible to configure the availability of command blocks to enable or to disable them (Steinmaurer, 2019). Figure 3.8 depicts sCool web application showing existing courses with a button to create new course. Figure 3.9 presents a view of the sCool web application showing an overview of concept learning mode tasks and an action

to create new concept-learning tasks. Figure 3.10 portrays sCool web application showing an overview of practical learning mode tasks and an action to create new practical tasks.

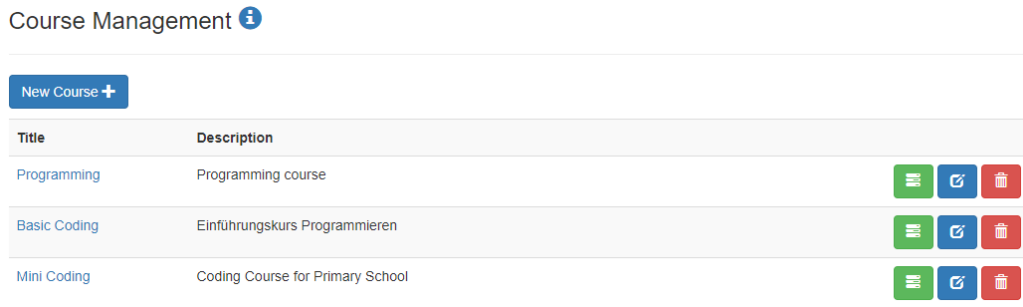


Figure 3.8: sCool Web application: overview of existing courses and new course create button (sCool version 3, year 2020.)

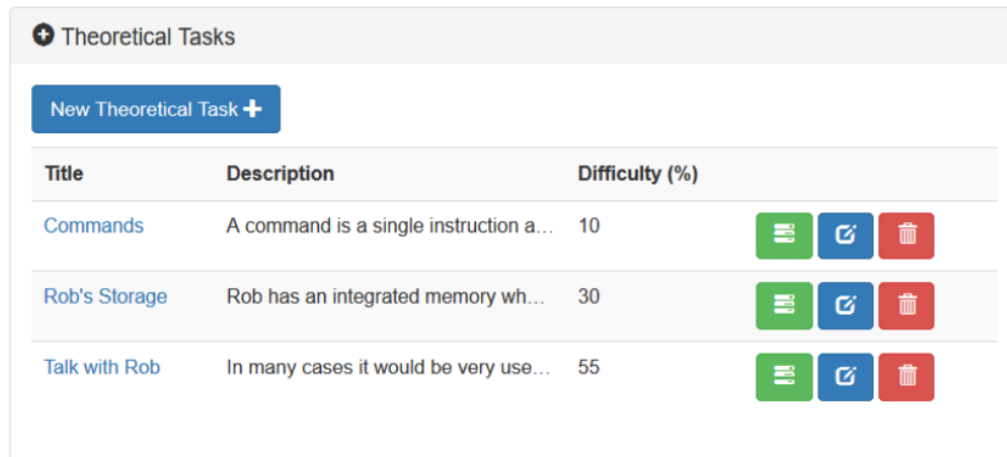
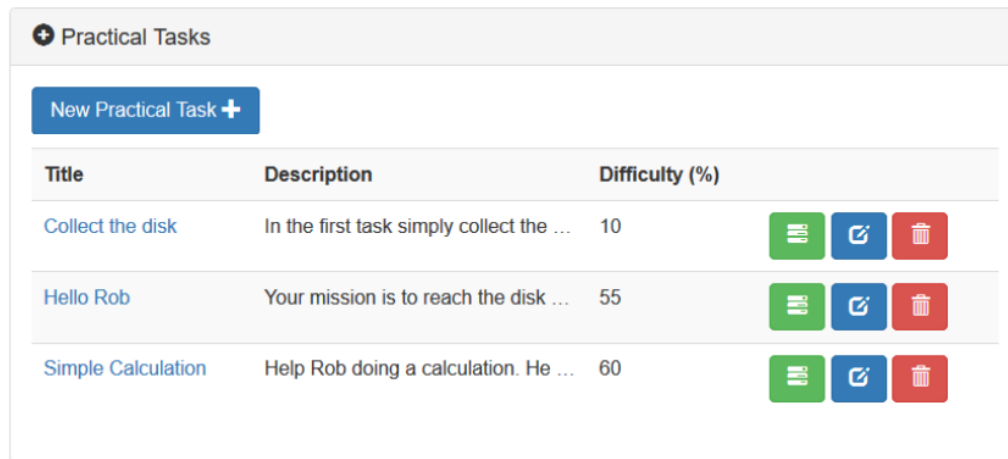


Figure 3.9: sCool Web application: Overview of Concept learning mode tasks of a course-skill (sCool version 3, year 2020.)



| Title | Description | Difficulty (%) | | | |
|--------------------|--|----------------|--|--|--|
| Collect the disk | In the first task simply collect the ... | 10 | | | |
| Hello Rob | Your mission is to reach the disk ... | 55 | | | |
| Simple Calculation | Help Rob doing a calculation. He ... | 60 | | | |

Figure 3.10: sCool Web application: Overview of Practice mode tasks of a course-skill (sCool version 3, year 2020.)

3.5 Summary

sCool consists of a mobile video game and a web platform. The mobile video game has concept-learning and practical parts. In the concept-learning part, players explore and gain new information related to programming concepts, followed by related questions. In the practical part, players use previously learned theoretical concepts to control a robot using Python instructions and collect disks.

The web platform is used to modify the highly adaptive learning content. It includes a course management tool using which learning content, and corresponding tasks, can be created and modified. It could be concluded that sCool lacks additional learning analytics abilities which could help various stakeholders make timely decisions for assisting students in need, or improving course content.

sCool architecture uses REST API to communicate between the mobile video game and server using HTTP methods and JSON objects to send and receive data.

Chapter 4

Requirements

This chapter discusses requirements for developing a learning analytics tool for the coding learning game sCool. The chapter also describes the requirements of various visualization techniques to assist users in simplifying information gathering from the tool.

4.1 Additional Features

sCool provides an immersive storyline of a space shuttle crashed on an unknown planet to keep its student players engaged. The mobile video game has concept-learning and practical parts which are seamlessly integrated. Students utilize the theoretical concepts learned in the concept-learning part in the practice mode, where they control a robot with programming instructions to collect disks for their space shuttle. The educators can create new courses using the web application and also modify their courses.

We also find that sCool could additionally provide learning analytics features to assist educators and game developers. Although the students can view an in-game overview of their course progress in the mobile game, which helps them keep track of their progress and make various strategic decisions for course completion, the educators lack a holistic view of their respective student's progress. This information

could help educators make timely decisions for assisting students in need or improving course content. Information such as coding concepts learned by students, student's game strategies for course completion, game interactions, points collected, and time spent completing game tasks could provide critical insights to the educators. As learning to program can be a daunting task for learners of any age (Lahtinen et al., 2005), timely assistance must be provided to any students in need.

With the sCool learning analytics application, we focus on providing an application that empowers educators and game developers with insights about the student's activities and analysis of courses, rather than making a decision tool for them (Dillenbourg et al., 2011). The empowered educators and game developers can then utilize their newfound knowledge to improve game content or assist students in need. We would further discuss the additional feature requirements in subsequent sections.

4.2 User Groups

Different stakeholders have various information needs based on their use cases (Börner et al., 2019). These needs must be understood to design compelling visualizations for communication and exploration.

We identify two types of stakeholder user groups who have supervisor roles as educators and sCool game developers or simply sCool game administrators. These user groups can be differentiated and classified into user groups of *educators* and *administrators*, respectively. An administrator can access all the data across all classes/learning activities happening on the sCool platform, whereas an educator has access to only their classes/learning activities. An administrator can gain additional insights by comparing various classes/learning activities, whereas an educator can only compare within their classes/learning activities.

4.3 Requirements

In this section, we discuss the functional and non-functional requirements of the sCool learning analytics application. We can group the requirements into functional

and non-functional requirements, about the system's functionality and behavior, and the system's constraints, respectively (Sommerville, 2010).

4.3.1 Functional Requirements

This subsection describes the services as functional requirements that the sCool learning analytics application must offer. Below is a list of the functional requirements to enable continuous improvement of the courses and learn through timely information that enhances decision making. The application should enable the educators to become proactive by delivering insights into learning experience of students in their classes/learning activities.

- **Class/Learning Activity Overview:** The application should provide educators with an overview of their classes/learning activities.
 - The application should enable educators to identify the practical and concept-learning tasks that students of their classes/learning activity solved.
 - The application should enable educators to identify the number of students who could acquire individual skills.
 - The educators should be able to view the code submitted by students of their class/learning activity.
 - The educators should also be able to identify the students who perform poorly and hence may need assistance with information such as course progress, tasks completed, points collected, code submissions, and errors in submitted code.
- **Programming Concepts learnt by the class:** The application should enable educators to identify the programming concepts learned by the class during the course/learning activity. The application should enable educators to identify the programming concepts learned during each of the tasks and, hence,

differentiate and ascertain the completion of each task's target goal for learning.

- **Student's overview:** The application should provide educators with an overview of student's game interactions and learning progress. The application should enable educators to view the students' final code submission and the coding concept learned by them in the course.
 - The educators should be able to gain insights into the student's interaction with the sCool game environment and game strategies undertaken to complete the courses.
 - The educators should be able to identify the coding concepts learned by the student.
 - The application should enable educators to identify students' progress, with information such as the time spent playing the game, tasks completed, skills acquired, and courses completed by the student.
 - The educators should be able to identify the type of errors in code submissions made by the student.
- **Analysis across classes/learning activities:** The application should enable educators to compare various classes/learning activities to identify any differences that require their attention. Some of the classes may be performing poorly compared to their peers. By identifying such discrepancies, educators could take timely actions to incentivize students for performance improvement or assist any students in need.
 - The application should enable educators to compare which classes performed better or worse compared to their peers.
 - The application should enable educators to identify the coding concepts learned by various classes through various task completion.

- **Statistical Data Analysis:** The application should also provide basic methods for statistical data analysis such as mean, standard deviation, and statistical distributions wherever necessary.

In addition to the above features, an administrator user profile could explore all the data across various courses, tasks, skills, and learning activities. An administrator could also compare the data across various courses, tasks, skills, and learning activities to gain insights.

4.3.2 Non-Functional Requirements

Below is a list of non-functional requirements which sCool learning analytics application should meet:

- **Usability:** A well-designed user interface should support educators and administrators in identifying and exploring data to find critical insights. An understandable and straightforward user interface enhances user's satisfaction with the system and motivates them to use it.
- **Reliability:** The application must be reliable and available all the time when needed.
- **Security:** The sCool environment contains student's data, and it is crucial to keep the system secure. A secure environment involves securing the game, web application, learning analytics application, and communications.
- **Modularity:** The codebase should be modular and clean. The modular structure should enable further development, maintenance, and collaboration easier.
- **Platform independent:** The application should be platform-independent to support users across various devices.

4.3.3 Data

The application must provide the information in simplistic ways such that an educator can gain knowledge with ease. Interactive data visualization tools could be used to make information easily available to the educators.

For simple statistics, bar charts, or scatter plots could be used to display game metrics of students game-related data. Graphs such as time series charts, or Gantt charts (Gantt, 1910) could be used to visualize student's game timeline and strategies undertaken for course completion. The focus of sCool learning analytics tool should be on lower-dimensional plots such as one-dimensional, two-dimensional, or three-dimensional plots to make the visualizations easy to understand for the stakeholders (Keim, 2002).

4.4 Conceptual Architecture

Based on the requirements discussed, a conceptual architecture model is designed. This conceptual architecture is a simplified representation of the sCool environment and the components involved. Figure 4.1 displays sCool system architecture with the integration of the sCool learning analytics application.

The sCool environment already consists of sCool mobile game and web application. The students play and learn using sCool mobile game, and the student's game-related data is sent to the web application to be stored in the database for further analysis. The web application provides the course content to the sCool mobile game.

The sCool learning analytics application connects to the database to access student's data. The sCool learning analytics application processes the player's game-related data from the database and provides meaningful information to the educators and sCool game developers (administrators). The learning analytics application also has a single sign-on redirect from the sCool web application to improve usability. The application examines game-related data of only those students who have explicitly agreed to allow their data for further analysis.

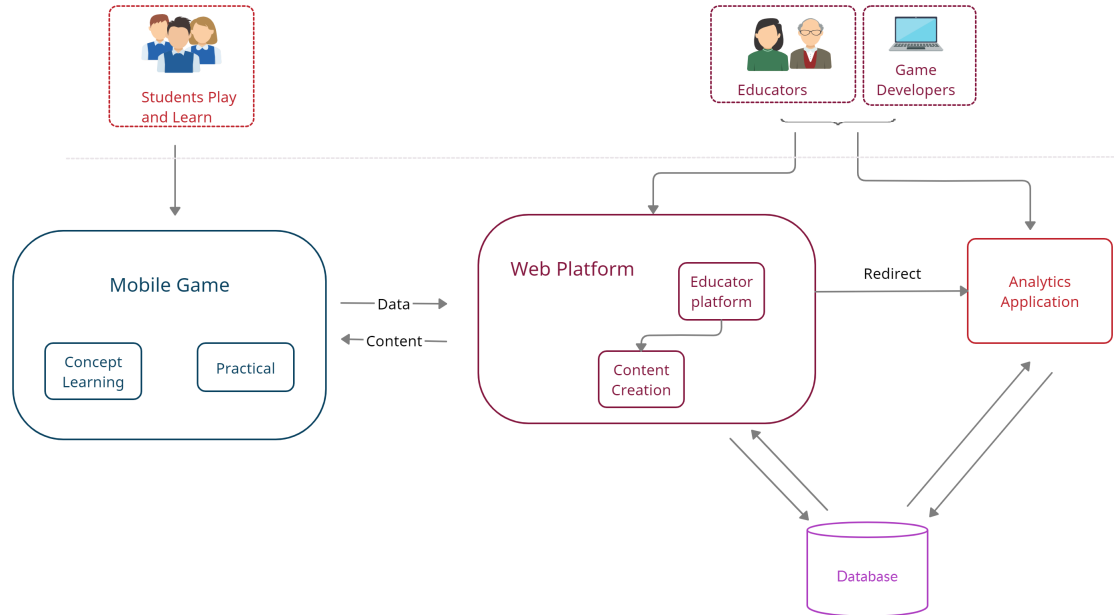


Figure 4.1: sCool - system architecture with the integration of the sCool learning analytics application. (adapted after Steinmaurer et al., 2019; Steinmaurer et al., 2020; Steinmaurer, 2019)

4.5 Summary

This chapter gives an overview of the requirements for developing a learning analytics application for the coding learning game sCool. The requirements are based on analysis and pre-evaluation of the coding learning game sCool, similar coding learning applications, and learning analytics tools. The application should empower educators and sCool game developers with insights about the students activities and progress in the courses on sCool game, rather than making a decision tool for them.

The focus is on supervisory stakeholders comprising of educators and sCool game developers. The users are classified as *educator* or *administrator* user groups. Educators have access to data of their own classes/learning activity, whereas administrators have access to data across all the classes/learning activities on sCool platform.

Significant functional requirements are of a platform-independent system, which provides insights about classes/learning activities and the included students to educators. This overview should enable educators to identify any students who may require assistance and improvement of course content.

The learning analytics application should have an easy-to-use and understandable user interface which is both reliable and secure. Usability is a central non-functional requirement. The codebase should be modular to enable further development and minimize maintenance efforts. It should also make collaboration work easier. The application should also be reliable and available whenever needed.

Chapter 5

Development

This chapter describes the development and implementation of a learning analytics application for the coding or computational skills learning game sCool. The concepts are based on the previously discussed requirements. The sCool system architecture with integration of the sCool learning analytics application is discussed. Additionally, the application’s development framework is also discussed. The chapter also shows key code listings and features that explain the development and functioning of the sCool learning analytics application. The chapter also describes the User Interface (UI) and related features of the sCool learning analytics application.

5.1 Architecture

Plotly’s Dash¹ is used as the application framework as it is an open-source full-stack framework for building web-based analytics application and it full-fills previously discussed requirements (Dash, 2021). Plotly’s Dash enables the creation of full-stack web applications with interactive visualizations combined with data analysis capabilities of Python² (Hossain et al., 2019). Since the framework enables creating a web application, it fulfills the requirements of being platform-independent, and hence users require only a browser to access the application from any device with

¹<https://plotly.com/dash/>

²<https://www.python.org/>

a web browser. Dash application uses Flask³ web server and React.js⁴ for creating interactive UI components.

Plotly⁵ has open-source graphing libraries in Python, JavaScript, and R to create dynamic, interactive charts and visuals. Using Python and R languages enables users to process complex data and models and then use the Plotly libraries to create interactive visuals. Plotly can also integrate with the Dash framework for building web-based analytic applications.

The framework provides dash-html-components⁶ and dash-core-components⁷ in addition to the main dash library. The dash-core-components provide a core set of components, written and maintained by the Dash team. The dash-html-components library provides components which mimic HTML elements.

Figure 5.1 displays a simplified application architecture depicting the various interacting components and libraries used in the application. The `index.py` defines the application's layout and defines the URL layout structure of the application. The `main.py` interacts with the sCool database. It establishes a connection with the sCool MySQL database to query player's game-related data. The queried data is then processed and a pandas⁸ DataFrame, which is a 2-dimensional data structure containing labeled axes (rows and columns), is used to store the data (Pandas DataFrame, 2021). The `login.py` handles user login POST⁹ requests and the application uses the Flask-Login module for user session management (Flask login, 2021). The `details.py` provides UI content to the main application layout. The components follow the Model-view-controller (MVC) pattern (Leff and Rayfield, 2001). When a user interacts with the components, the controller handles the user inputs manipulating information of the Model and updates the View.

The application uses Docker¹⁰ for packaging the application and its dependencies

³<https://flask.palletsprojects.com/en/1.1.x/>

⁴<https://reactjs.org/>

⁵<https://plotly.com/>

⁶<https://dash.plotly.com/dash-html-components>

⁷<https://dash.plotly.com/dash-core-components>

⁸<https://pandas.pydata.org/docs/index.html>

⁹<https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html>

¹⁰<https://www.docker.com/>

in a virtual Docker container¹¹. The Docker container image packages code and all the related dependencies so the application runs reliably on different computing environment. The Docker container image could then be run on any Windows, or Linux OS.

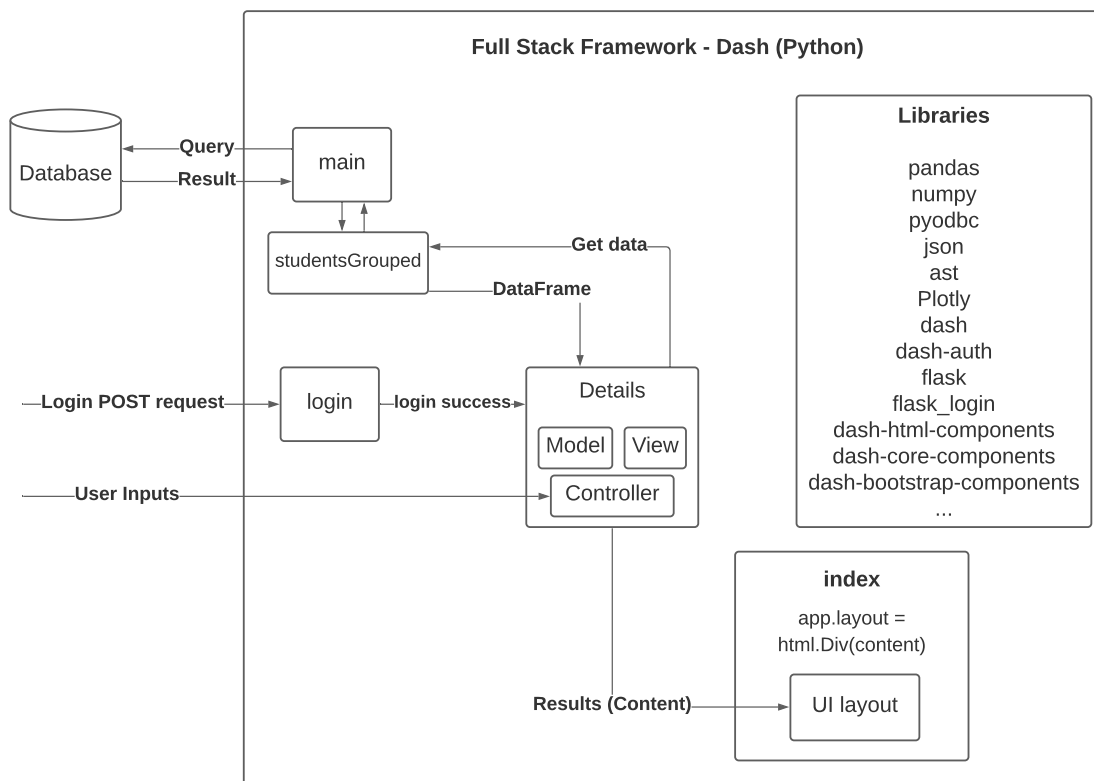


Figure 5.1: Simplified Architecture

5.2 Implementation

This section covers the implementation details of the sCool learning analytics application using the Dash open source framework as discussed in previous sections.

¹¹<https://www.docker.com/resources/what-container>

5.2.1 Connecting with Database

The sCool learning analytics application connects with the sCool Database to get students game-based data stored in the database. *pyodbc*¹² is an open-source Python module and is used for establishing a connection to sCool’s MySQL database. *pyodbc* is an open-source Python module that makes accessing ODBC (ODBC overview, 2021) databases simple.

Listing 5.1 shows a simple python program to enable connecting to a server and database.

```
1 import pyodbc
2
3 conn = pyodbc.connect('Driver={<AddDriverNameHere>; '
4                       'Server=<AddServerNameHere>; '
5                       'Database=<AddDatabaseNameHere>; '
6                       'Uid=<AddUserNameIdHere>; '
7                       'Pwd=<AddPasswordHere>; '
8                       'Port=<AddPortHere>; '
9                       'Trusted_Connection=<
  AddIsTrustedConnectionHere>;')
```

Listing 5.1: Connecting to database using pyodbc

5.2.2 Python Code Parser for Code Concepts Used

Listing 5.2 shows a simple python class *PythonCodeParser* to parse Python code submitted by students using Python *ast* module (Python ast, 2021). The class *PythonCodeParser* creates an AST from a submitted python program and finds the programming concepts used by the students. The class *PythonCodeParser* analyzes code solutions submitted by players and finds coding concepts used/learned by the students. Python’s in-built function *getattr*¹³ can be used to get the named attribute of an object. The application uses a similar class with additional properties as

¹²<https://pypi.org/project/pyodbc/>

¹³<https://docs.python.org/3/library/functions.html#getattr>

PythonCodeParser to analyze programming solutions submitted by students and find coding constructs used by students in the submitted programming solutions.

```
1 import ast
2
3 class PythonCodeParser:
4     def __init__(self, expr):
5         self.expr = expr
6         self.tree = ast.parse(self.expr, mode="exec")
7
8     # programming concepts used in the code - does program has
9     # arithmetic expressions (Example: addition, subtraction, division
10    )
11    def hasExpressionsArithmetic(self):
12        for node in [n for n in ast.walk(self.tree)]:
13            if isinstance(node, ( ast.Add, ast.Sub, ast.Mult,
14            ast.Div, ast.FloorDiv, ast.Mod, ast.Pow )):
15                return True
16        return False
17
18    # programming concepts used in the code - does program has
19    # Boolean expressions (Example: Boolean And, Boolean or)
20    def hasExpressionsBool(self):
21        for node in [n for n in ast.walk(self.tree)]:
22            if isinstance(node, ( ast.And, ast.Or )):
23                return True
24        return False
25
26    # programming concepts used in the code - does program has Loops
27    # (Example: for and while loop)
28    def hasLoop(self):
29        for node in [n for n in ast.walk(self.tree)]:
30            if isinstance(node, (ast.For, ast.While)):
31                return True
32        return False
33
34 codeConcepts = PythonCodeParser(
35     'print("a simple python program") \n'
```

```
31     + 'a = 1 + 2' )
32
33 print('Does the python code has a arithmetic expressions?', int(
34     codeConcepts.hasExpressionsArithmetic() ))
35 codeConceptsParsedFunctionLoop = getattr(codeConcepts, 'hasLoop',
36     None)
37 print('Does the python code has a loop?', int(
38     codeConceptsParsedFunctionLoop() ))
```

Listing 5.2: Parsing a python program to create AST for examining coding concepts used

5.2.3 JSON Parser

Listing 5.3 shows a simple python code to parse JSON as a string using Python *json* module (Python json, 2021). Some of the player's game-related data, such as all the code executed by players before the final submission, dragged code block options, deleted code, and tabs switched, are collected and stored as JSON as a string on the database. Hence, it is important to extract relevant features from the JSON stored as a string for further analysis. Before the final submission, the code executed by players provides information about the various errors that occurred when students attempted programming tasks.

```
1 import json
2 def read_json(json_data):
3     if (type(json_data) == str): # For strings
4         return json.loads(json_data)
```

Listing 5.3: Parsing a JSON string

5.2.4 Data Processing

Figure 5.2 displays a diagrammatic representation of steps for generation of output in the application. The application uses *pandas*¹⁴ open source library as it provides high-performance, easy-to-use data structures, and data analysis tools for the Python programming language (Pandas, 2021). The application connects with the sCool MySQL database to receive player’s game-related data. The player’s game-related data is stored in a pandas DataFrame which is a 2-dimensional data structure containing labeled axes (rows and columns) (Pandas DataFrame, 2021). The JSON as string features are processed, and the features within them are extracted since the JSON as string features contain information such as code runs and tabs switched by players. The programming solutions of players are analyzed to find the coding concepts used. The resultant features are merged with the DataFrame. The data is cleaned by removing duplicate features and replacing NaN (Not a Number) values with zeros (NaN, 2021). The data is grouped by LearningActivityId (Pandas groupby, 2021). Group by LearningActivityId makes it easier to get information relating to a selected Learning Activity. When a user action provides a selected Learning Activity, the related data is selected from the grouped data and displayed to an end-user with relevant data visualization techniques. Listing 5.4 shows a simple python code with the above discussed steps.

¹⁴<https://pandas.pydata.org/docs/index.html>

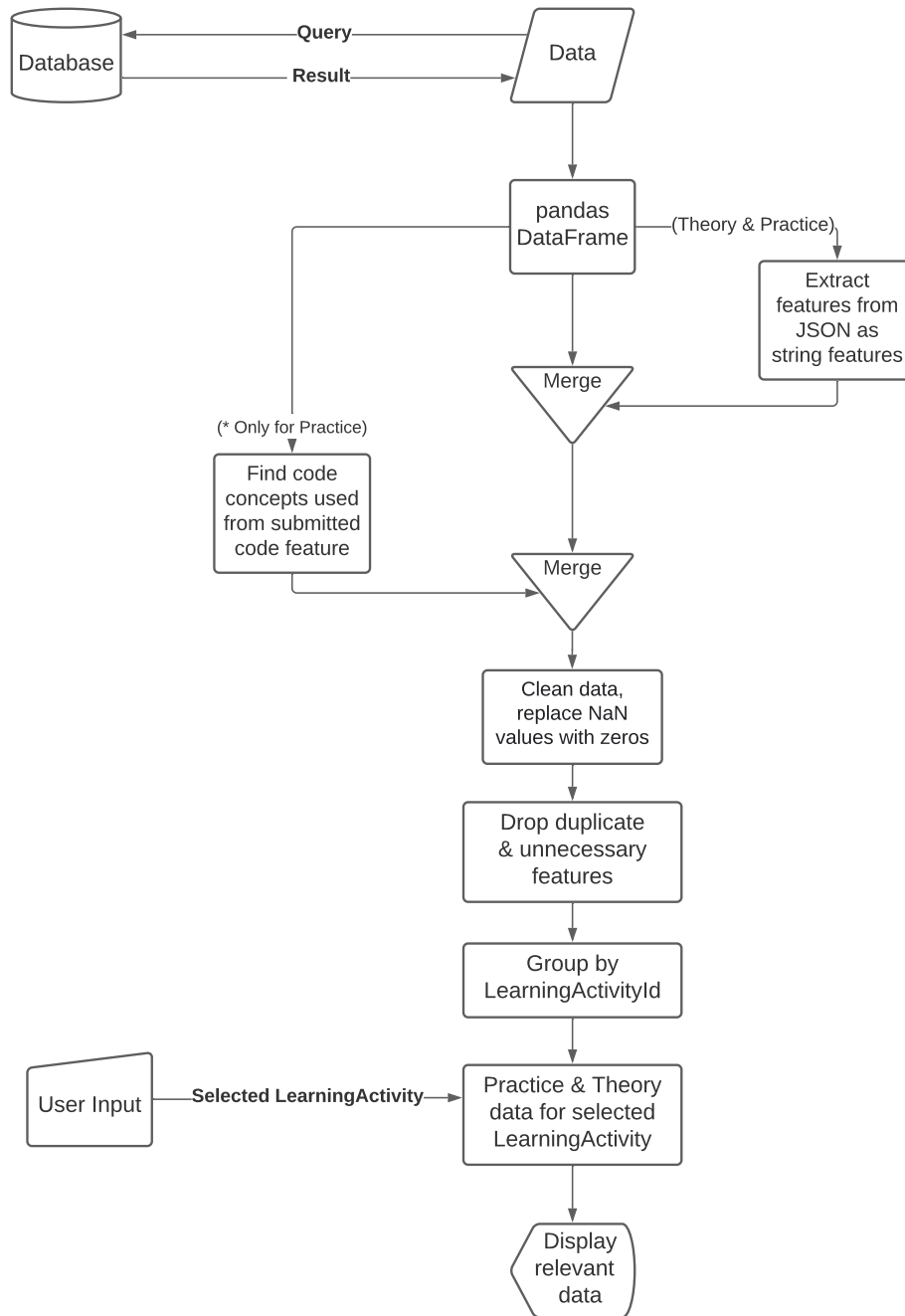


Figure 5.2: Steps of data processing

```
1 #Query Practice data for Student's who have given consent for data
  processing
2 dfPractice          = pd.read_sql_query(<MySQL query string:
  SELECT Students Practice Data WHERE students.IsConsentGiven IS
  True> , conn)
3
4 #Extract data from JSON as string Features, e.g. feature 'Runs'
5 dfRuns = getDataframeFromJsonFeature('Runs', dfPractice)
6
7 #Number of times code executions ('Run') had errors
8 runsErrorCount = dfRuns[dfRuns.RunsError == True].
  PracticeStatisticsId.value_counts()
9 countDict = runsErrorCount.to_dict()
10
11 #Create feature for number of time errors occurred in code
  executions
12 dfPlayerStrategyPractice['runsErrorCount'] =
  dfPlayerStrategyPractice['PracticeStatisticsId'].map(countDict)
13
14 #Code concepts used
15 conceptFeaturesLines = extractConceptFeaturesFromCodeLines(
  dfPractice, 'Code')
16
17 #Merge extracted features
18 dfPractice = dfPractice.merge(conceptFeaturesLines, how='left')
19
20 #remove all features which have no meaning
21 dfPractice.drop(['<listOfFeature2Drop>'], inplace=True, axis=1)
22
23 #Replace NaN with zero
24 dfPractice.fillna(0, inplace=True)
25
26 #Group data by LearningActivityId - to make it easier for selection
  by LearningActivityId
27 dfGrouped = dfPractice.groupby( [df['LearningActivityId']] )
28
```

```
29 #Get Learning Activity data
30 learningActivityData = dfGrouped.get_group('<
    SelectedLearningActivityId>')
31
32 #Display data to users using relevant techniques (such as a table)
33 createTable(learningActivityData)
```

Listing 5.4: Players game-related data processing step

5.2.5 Login Redirect and User Session Management

Listing 5.5 shows a HTML5 code to redirect user to the sCool learning analytics application eliminating requirement of an additional manual login. A HTML5 form¹⁵ is submitted with POST¹⁶ method using logged in users security stamp. The POST method is a common HTTP¹⁷ method where the form data is sent as the request body. The login redirect form can be integrated into the existing sCool web-application for content creation.

```
1 <!DOCTYPE html >
2 <html >
3     <head >
4         <title>Login Redirect</title >
5     </head >
6     <body >
7         <form action="<applicationURL >/login?securityStamp=<
    userSecurityStamp >" method="post">
8             <input type="submit" value="Go to sCool learning
    analytics application">
9         </form >
10    </body >
11 </html >
```

Listing 5.5: User redirect form

¹⁵<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form>

¹⁶<https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html>

¹⁷<https://www.w3.org/Protocols/rfc2616/rfc2616.html>

The sCool learning analytics application uses the Flask-Login module for user session management (Flask login, 2021). Flask-Login provides user session management for Flask with functions such as logging in, logging out, and retaining or remembering a user's session over some time (Flask login, 2021). The Flask-Login module stores the active user ID in the session and enables restricting the views to logged-in or logged-out users. The Flask-Login module enables redirect from the sCool web application to the sCool learning analytics application without requiring users to provide username and password again.

The sCool learning analytics application processes the form information, and the user's session is created and managed using the Flask-login module. Listing 5.6 shows the Python code for processing the login request and creating a user session.

```
1 from flask import Flask, request, redirect
2 from flask_login import logout_user, current_user, LoginManager,
   UserMixin
3 from dash.dependencies import Input, Output, State
4
5 class User(UserMixin):
6     def __init__(self, name, Id, active=True, isAdmin=False,
   securityStamp='', isAuthenticated=False):
7         self.name = name
8         self.id = Id
9         self.active = active
10        self.isAdmin = isAdmin
11        self.securityStamp = securityStamp
12        self.authenticated = isAuthenticated
13
14    def is_active(self):
15        return self.active
16
17    def is_authenticated(self):
18        """Return True if the user is authenticated."""
19        return self.authenticated
20
21    def is_admin(self):
22        return self.isAdmin
```

```
23
24 # login POST method request handler
25 @server.route('/login', methods=['POST'])
26 def login():
27     if request.method == 'POST':
28         # verify the security stamp with additional checks
29         if request.args.get('securityStamp'):
30             user = User('<UserName>', '<UserId>', active='<isActive>
31             ', isAdmin='<IsAdmin>', securityStamp='<SecurityStamp>',
32             isAuthenticated=True )
33             login_user(user)
34
35 # to find if user is logged in
36 def isAUserLoggedIn():
37     if current_user and current_user.is_authenticated:
38         return True
39     else:
40         return False
41 isAUserLoggedIn()
```

Listing 5.6: User login POST request handler

5.2.6 Model-View-Controller

Listing 5.7 shows a simple dash application Python code for dash components following Model-view-controller (MVC) design pattern (Leff and Rayfield, 2001). The users see the View, which contains the layout. The Model only contains the data and does not define any logic or user behavior. The controller contains the logic, handles various user interactions with the components, and manipulates the Model and View information.

```
1 import dash_core_components as dcc
2 import dash_html_components as html
3 from dash.dependencies import Input, Output
4 import pandas as pd
5
```

```
6 # Model
7 data = {
8     "main": [111, 3333, 8888],
9     "details": [50, 40, 45]
10 }
11 df = pd.DataFrame(data)
12
13 # View
14 layout = html.Div([
15     html.H1("This is a View"),
16     html.Div(dcc.Input(id='input-from-user', type='text')),
17     html.Button('SubmitUserInput', id='buttonSubmit'),
18
19     html.Div(id='output-user-action',
20             children= 'user action results' )
21     html.Div(id='output-user-action-result',
22             children= df['main'][0] )
23 ])
24
25 # Controller handling user actions
26 @app.callback(
27     dash.dependencies.Output('output-user-action', 'children'),
28     [dash.dependencies.Input('buttonSubmit', 'n_clicks')],
29     [dash.dependencies.State('input-from-user', 'value')])
30 def on_user_interaction(n_clicks, value):
31     return 'The User performed an action'
32
33 # Controller handling user actions
34 @app.callback(
35     dash.dependencies.Output('output-user-action-result', 'children'
36     ),
37     [dash.dependencies.Input('buttonSubmit', 'n_clicks')],
38     [dash.dependencies.State('input-from-user', 'value')])
39 def on_user_interaction(n_clicks, value):
40     return df['details'][int(n_clicks)]
```

Listing 5.7: Model-View-Controller

5.2.7 Data Visualization Techniques

The sCool learning analytics application uses various visualization techniques for communicating data to users. Listing 5.8 shows the Python code for generating a HTML div¹⁸ with label and value. The label and value are highlighted using CSS styles. The dash-html-components¹⁹ library is used to compose layout using Python structures which gets converted to HTML in the application.

```
1 import pandas as pd
2 import dash_html_components as html
3 '''
4 CSS styles in appStyle.css
5 .c-card.c-card-small {
6     margin: 0.5rem;
7     padding: 0.5rem;
8 }
9 .c-card.c-card-small .card_label {
10     font-size: 1rem;
11 }
12 .c-card.c-card-small .card_value {
13     font-weight: bold;
14     font-size: 1.3rem;
15 }
16 '''
17
18 def generateCardBase(label, value, classes = ""):
19     return html.Div(
20         [
21             html.Span(
22                 children = [ value ],
23                 className="card_value"
24             ),
25             html.P(
26                 label,
27                 className="card_label"
```

¹⁸<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/div>

¹⁹<https://dash.plotly.com/dash-html-components>


```

28         ),
29     ],
30     className="c-card " + classes,
31 )
32
33 generateCardBase( [ <FeatureLabel> ],
34     playersDataFrame[<FeatureName>].sum() ,
35     classes = "c-card-small" )

```

Listing 5.8: Data visualization highlighting key numbers

Listing 5.9 shows the Python code for generating a horizontal bar charts²⁰ using the Plotly express²¹ module. The values are sorted in descending order before displaying using pandas dataframe `sort_values`²² function. Horizontal bar charts are used to show information as a simple visualization technique for easy interpretation.

```

1 import pandas as pd
2 import plotly.express as px
3 import dash_core_components as dcc
4
5 def getSimpleFeaturePlot(df, featureX, featureY, title, hoverData):
6     graphs = []
7
8     # sort values in descending order
9     df = df.sort_values(    by = [ featureX ]    )
10
11     fig = px.bar( df
12         , x            = featureX
13         , y            = featureY
14         , title        = title
15         , orientation  = 'h'
16         , hover_data   = hoverData
17     )
18
19     graphs.append(

```

²⁰<https://plotly.com/python/bar-charts/>

²¹<https://plotly.com/python/plotly-express/>

²²https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.sort_values.html

```
20     dcc.Graph(  
21         figure= fig  
22     )  
23 )  
24  
25 simpleHorizontalBarPlots = getSimpleFeaturePlot('<playersDataFrame>'  
26     , '<featureXaxis>', '<featureYaxis>',  
27         '<This is the title>',  
        ['<list of features in the playersDataFrame to show  
        on mouseover interaction - hover data>'])
```

Listing 5.9: Data visualization showing players data using a horizontal bar chart

5.3 User Interface

Figure 5.3 depicts the application’s User Interface (UI) which is divided into the following central regions:

- Left section (1)
 - Navigation section (5)
- Main section (2)
 - Selection of Learning Activity/Class section (3)
 - Content and details section (4)

The left region contains the navigation bar, application logo, and settings action. The right region contains all the content and details. The right region contains a selection of learning activities/classes, followed by their content and details. The user interface follows material design guidelines.

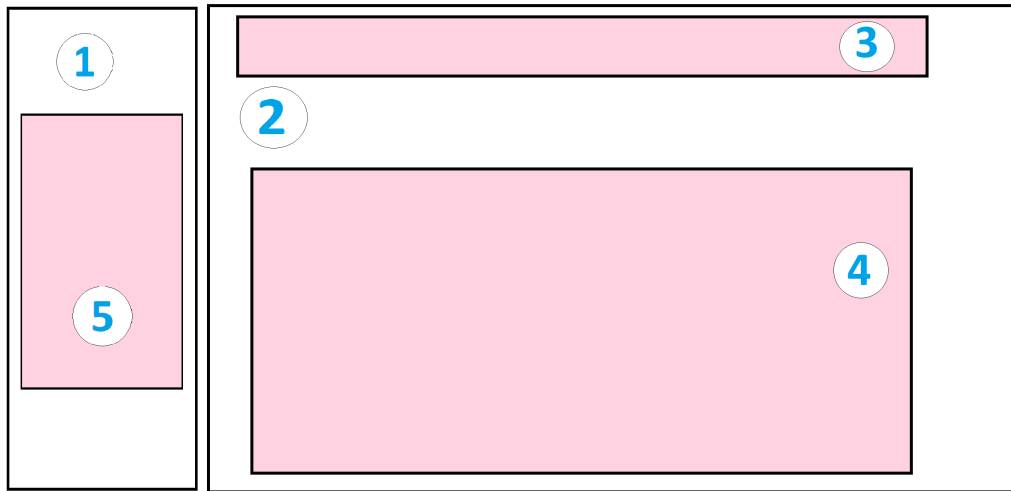


Figure 5.3: Overview of user interface

The user interface is designed to be responsive and adaptive to various screen sizes. The user interface should be intuitive and straightforward as it should be easy to understand the features provided by the application and learn to use them.

The application has various sections as described below, with access for an Educator (referenced with *E* below) and Administrator (referenced with *A* below):

- Details (A, E): detailed overview of a learning activity/class. Users can gain an overview of a learning activity/class with information such as the number of student participants, total game time spent, task-wise completion rate, student-wise tasks completed, concepts used by students of the class, and points collected.
- Students (A, E): detailed overview of a student of a learning activity/class. Users can overview a student's game interactions with information such as game time, time spent completing practical and concept-learning tasks, coding concepts learned, course progress, and student's timeline.
- Custom (A, E): users can create custom visualization by selecting features, graph type, and other information.

- Game Data (A): users can explore sCool’s entire game data by creating custom visualizations.
- Groups (A): users can compare the overall performance of various classes/learning activities compared to one another.

5.4 Features

The sCool learning analytics application strives to present users with an interactive, informative and straightforward UI. In the following subsections, we describe features accessible to both the user groups of educators and administrators (sCool game developers). An administrator has access to all features of an educator.

5.4.1 Educator User Group

The educator user group has access to student’s game-related data of their classes/learning activities. The educators can select a class/learning activity from a list of their classes/learning activities. The educators can also gain insights into individual student’s game-related data by selecting them from a list of students. This subsection describes the information and features available to an educator user. This subsection describes the application sections and the data within these sections, which an educator user can access.

5.4.1.1 Learning Activity/Class Details Section

Figure 5.4 provides an overview of the UI for an educator user group. In the left region, the user can navigate various sections using the navigation bar (A). At first, the user selects their learning activity/class in the central region, which they would like to gain insights about (B). This provides the user with an overview of the learning activity/class in the content section (C).

5.4.1.2 Student Details Section

An educator can navigate to the Students section to gain insights about a student. Figure 5.5 displays the Students details section. An educator can select students by their game username in a dropdown (D). This results in an overview of the selected student in the content section (C), displaying information like game time, time spent in practical and concept-learning sections, points earned, and the number of coins collected. The course progress card of the selected student is also depicted (E). The educator can also see completed tasks and courses by the student in the course progress card (E).

Figure 5.6 displays Students game interaction details. The student's game interaction depicts the tasks performed by the student in the game with respect to time (F). On mouseover (mouseover event, 2021) user interaction, additional details can be seen about the specific task performed and the time.

Figure 5.7 displays an overview of student's game timeline chart with mouseover (mouseover event, 2021) user interaction providing additional details concerning student's activities at a particular time. The student's timeline shows the tasks overtaken by the student in a time series Gantt chart (G). Users can observe the various strategies taken by students for course completion. With mouseover interaction's users can find information such as the task attempted and result.

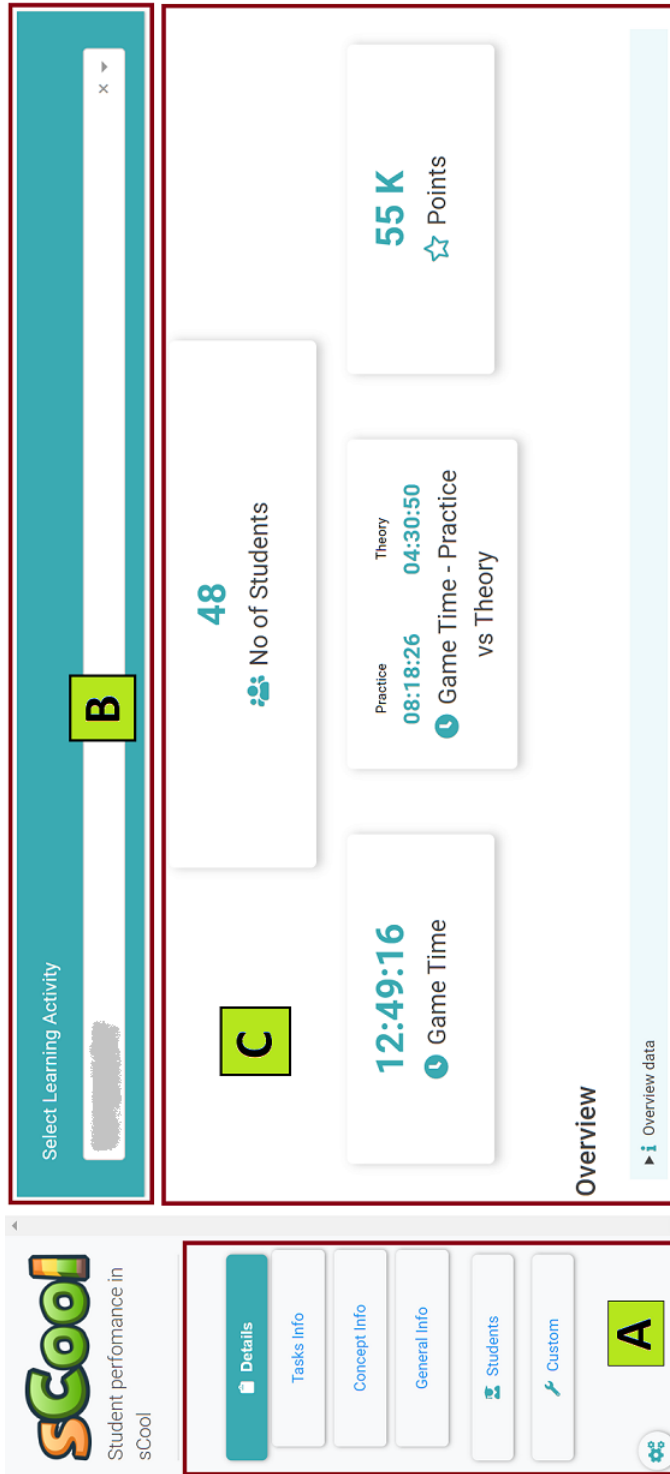


Figure 5.4: Overview of details section

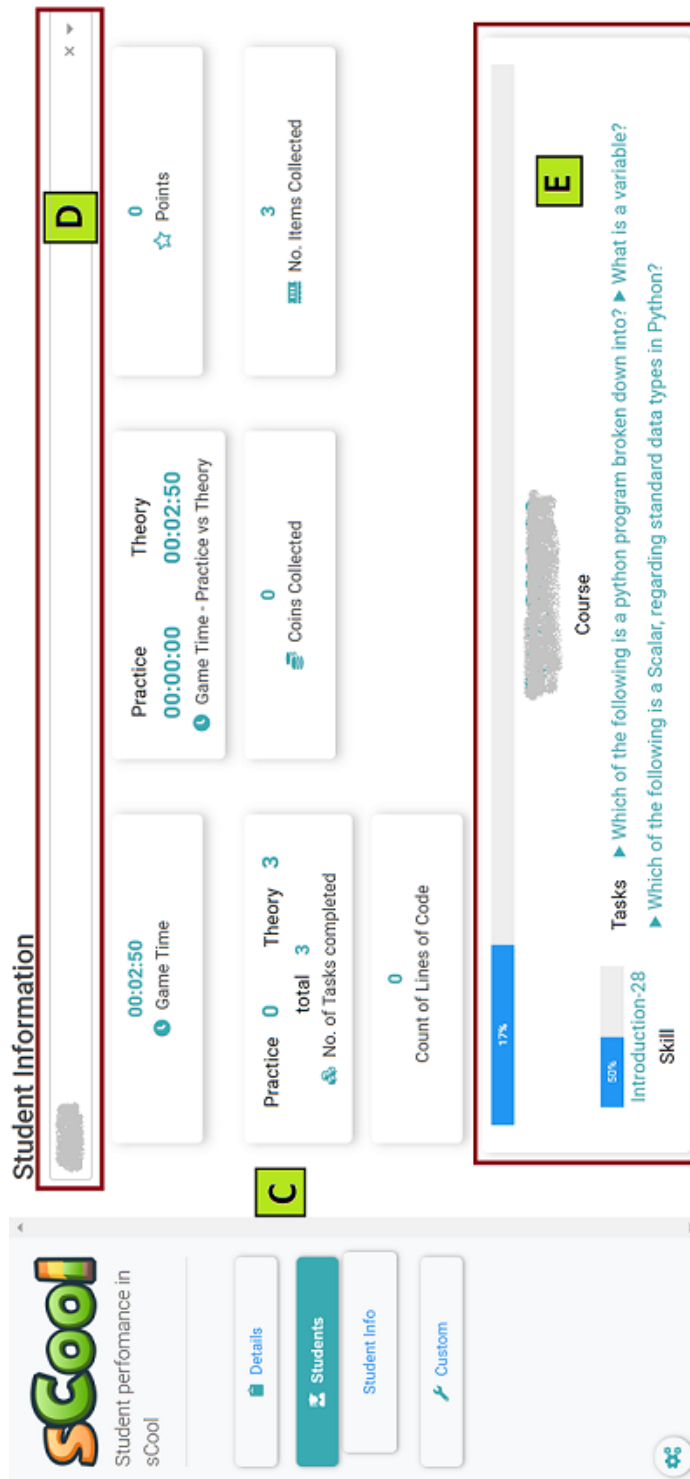


Figure 5.5: Overview of students section

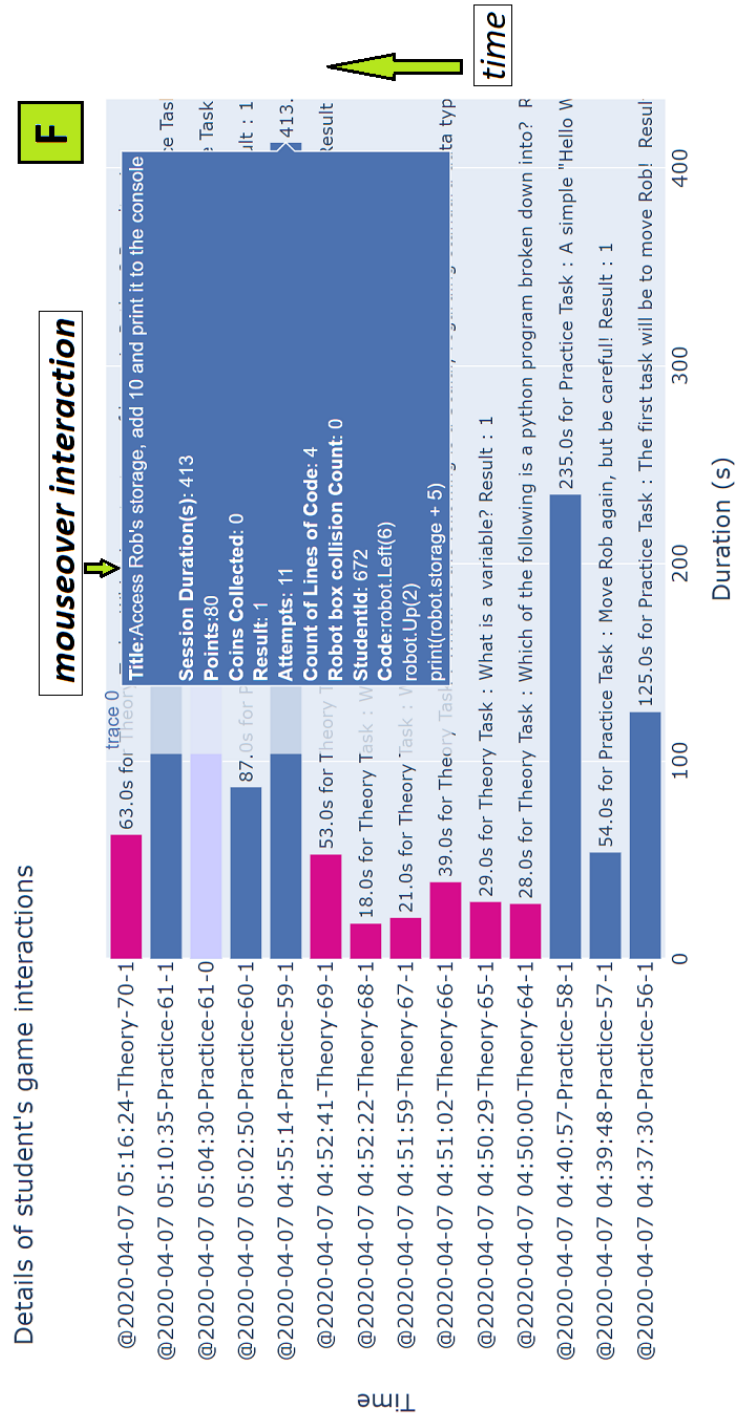


Figure 5.6: Overview of students game interactions

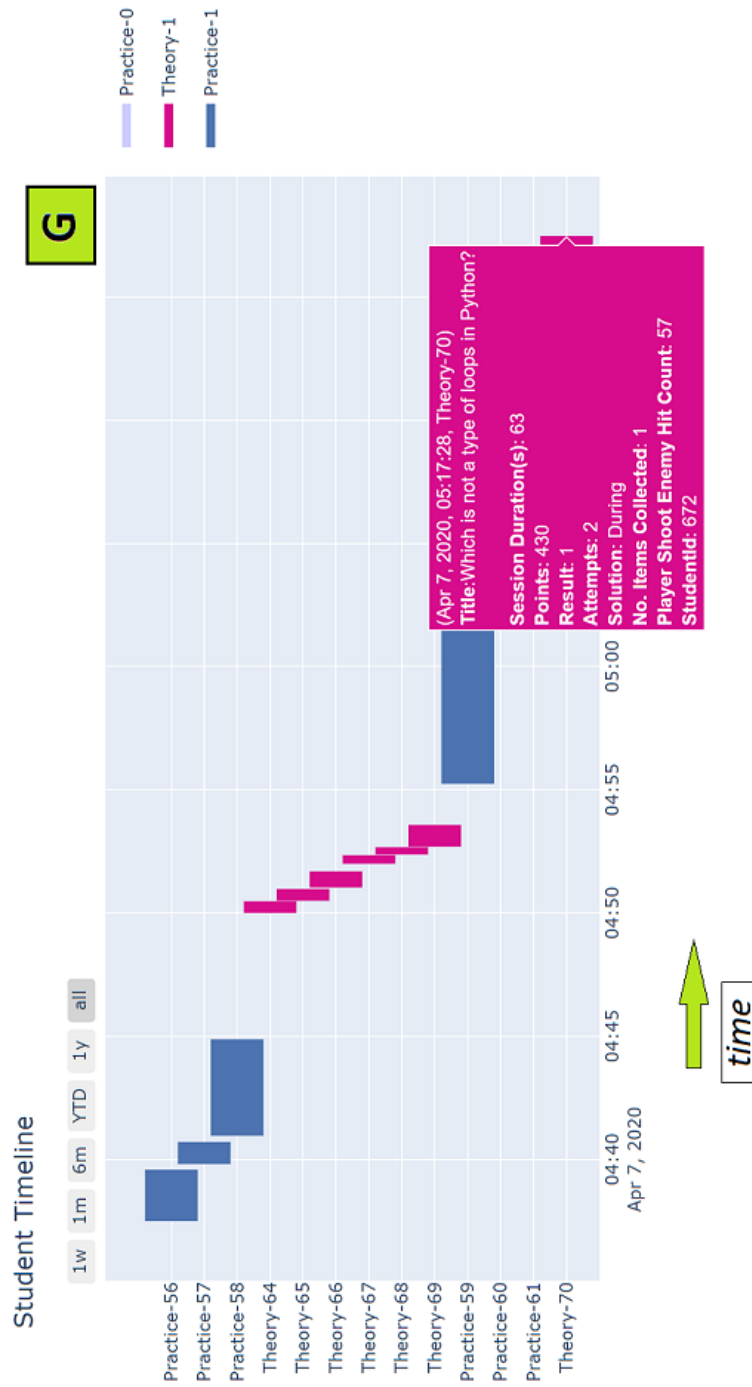


Figure 5.7: Overview of students game timeline chart with mouseover user interaction

5.4.1.3 Custom Details Section

An educator can navigate to the Custom section to gain any additional insights besides the Details and Student section, such as errors in code, syntax errors in code, average lines of code, and coins collected. Users can explore data by creating simple visualizations, data tables, or distribution plots. The custom section consists of a plot generation form and content section. Figure 5.8 displays the Custom section. Users can select the primary data they would like to gain insights about, for example, students, tasks, skills, course, and task type (H). Users can then select the x-axis (I), and y-axis (J) features for various visualization chart types (K). Currently, the users can generate visualization charts like bar, scatter, bubble, line, or data tables (K). Users can also select sub-grouping in the data (M). Finally, the users can click the *Add Plot* action to generate a new visualization. Figure 5.9 displays the newly generated plot showing details of the student’s session duration. Every new visualization is appended at the top of the content section. Hence the users can also see the previously generated plots and compare them to the newly generated plots or use them together for analysis.

Figure 5.10 displays a plot of number of errors that occurred in code submitted by students for completion of course tasks on the sCool learning game. The figure shows that most of the errors occurred by students in a specific practice task. Educators or game developers could then analyze the task and identify why students face many errors in the task. The task could be player’s first python program, difficult for students, or have unclear instructions. On further investigations, the educators or game developers could come up with improvements if required.

Figure 5.11 displays a plot of errors in player’s code submissions providing information such as the students who faced errors in completing coding tasks.

Figure 5.12 depicts the custom plot of the number of times players switched to read a task description for each of the tasks in a sample class/learning activity. The task descriptions may be unclear and hard to understand for players. The task descriptions where players had to read them multiple times could be rechecked to ensure that the instructions are detailed and novice learners could also follow them.

The screenshot shows the 'Custom' section of the sCool interface. At the top left, the 'Student performance in sCool' header is visible. Below it are navigation buttons for 'Details', 'Students', 'Custom' (highlighted), and another 'Custom' button. The main form area is titled 'Select Learning Activity' and contains a search bar with 'RMIT Martin' entered. Below this are two sections: 'Select Group By' with radio buttons for 'Student' (selected), 'Task', 'Skill', 'Course', and 'Task Type'; and 'Select Sub Group By' with a dropdown menu set to 'Select features'. A 'Session Duration(s)' dropdown is also present. The 'Select Features' section includes a 'Session Duration(s)' dropdown, a 'Type' section with radio buttons for 'Bar' (selected), 'Scatter', 'Bubble', 'Line', and 'Table', and a 'Distribution' section with radio buttons for 'Horizontal (x-axis)' (selected), 'Vertical (y-axis)', 'mean', 'std', and 'median'. A 'Distribution' section also has radio buttons for 'All details', 'mean', 'std', and 'median'. At the bottom, there is a 'Select features' dropdown, an '+ Add Plot' button, and a blue double-arrow button.

Figure 5.8: Overview of Custom section form

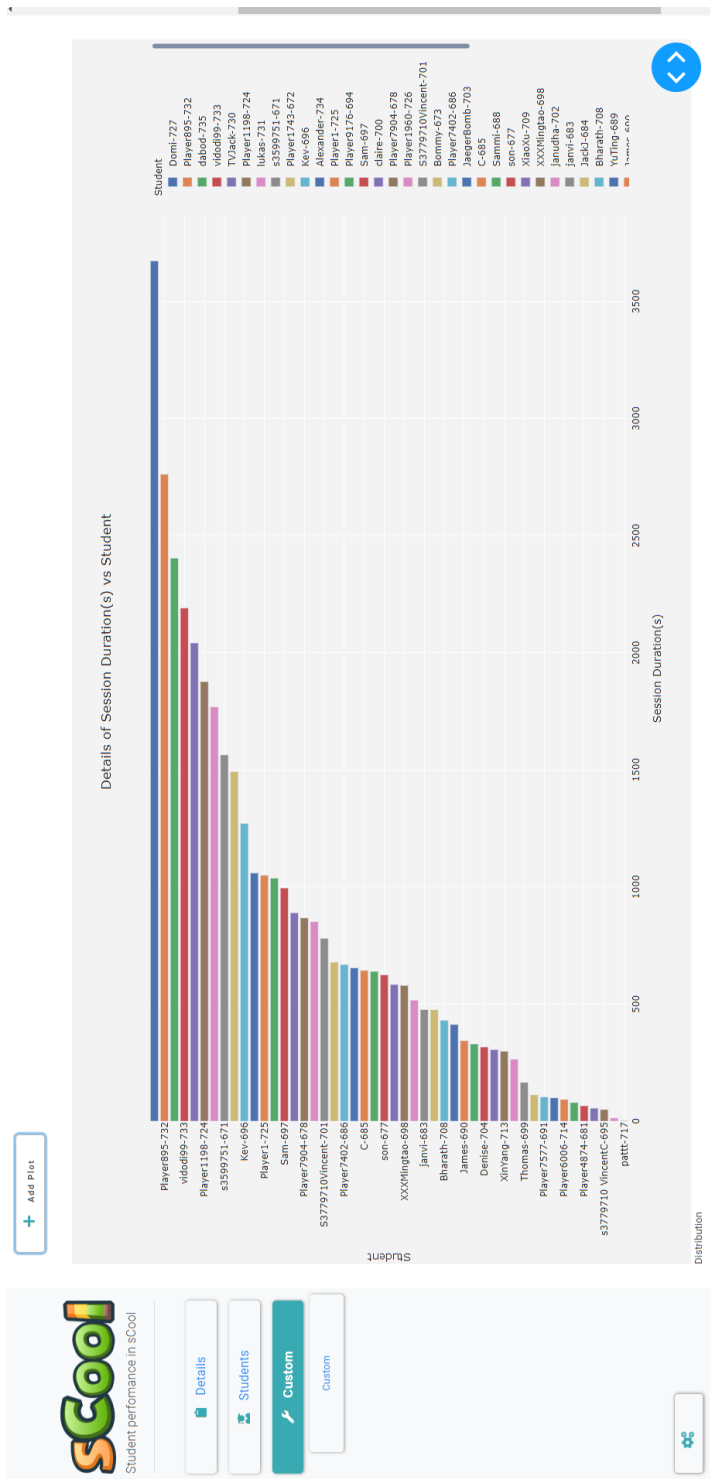


Figure 5.9: Example of a custom plot

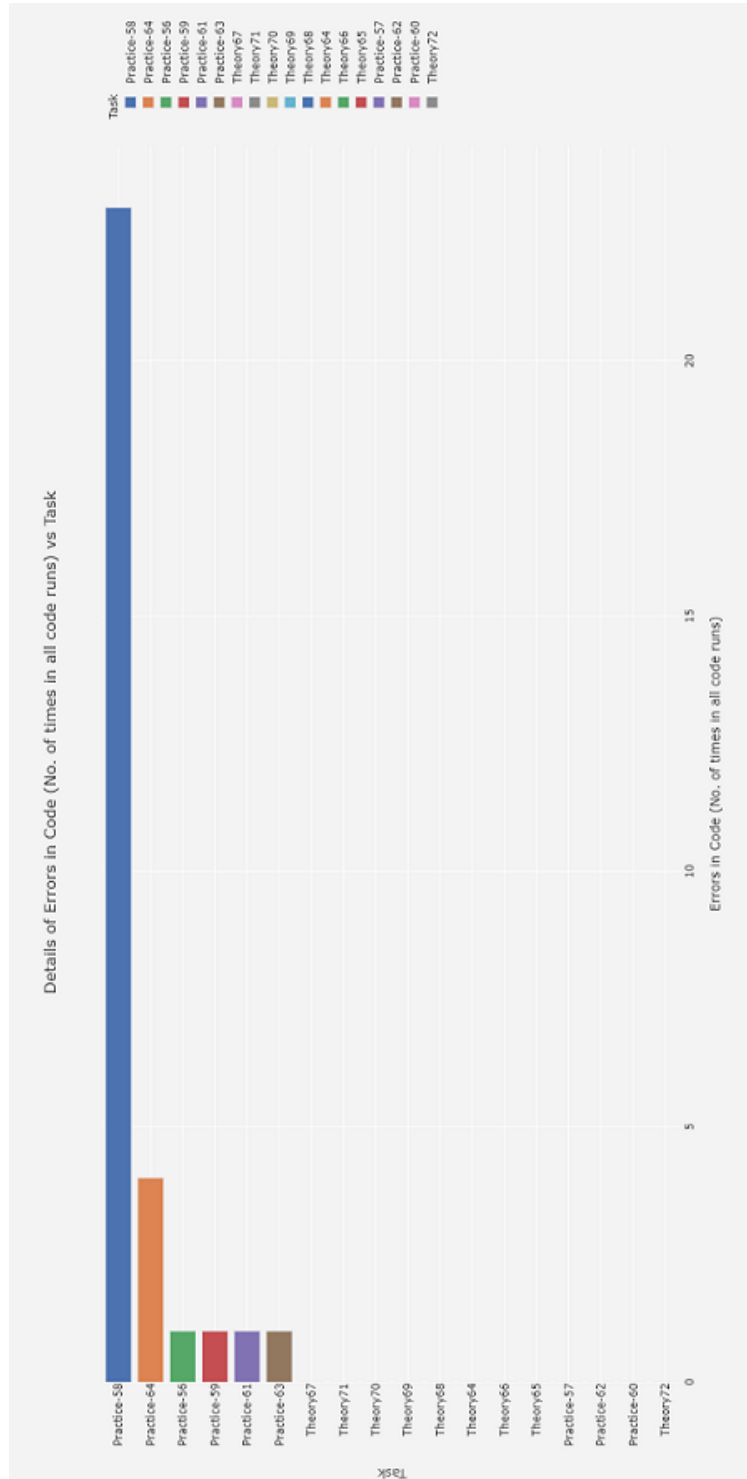


Figure 5.10: A custom plot of the number of errors that occurred in program solutions submitted by students for course tasks showing that students had many errors in a particular practice task

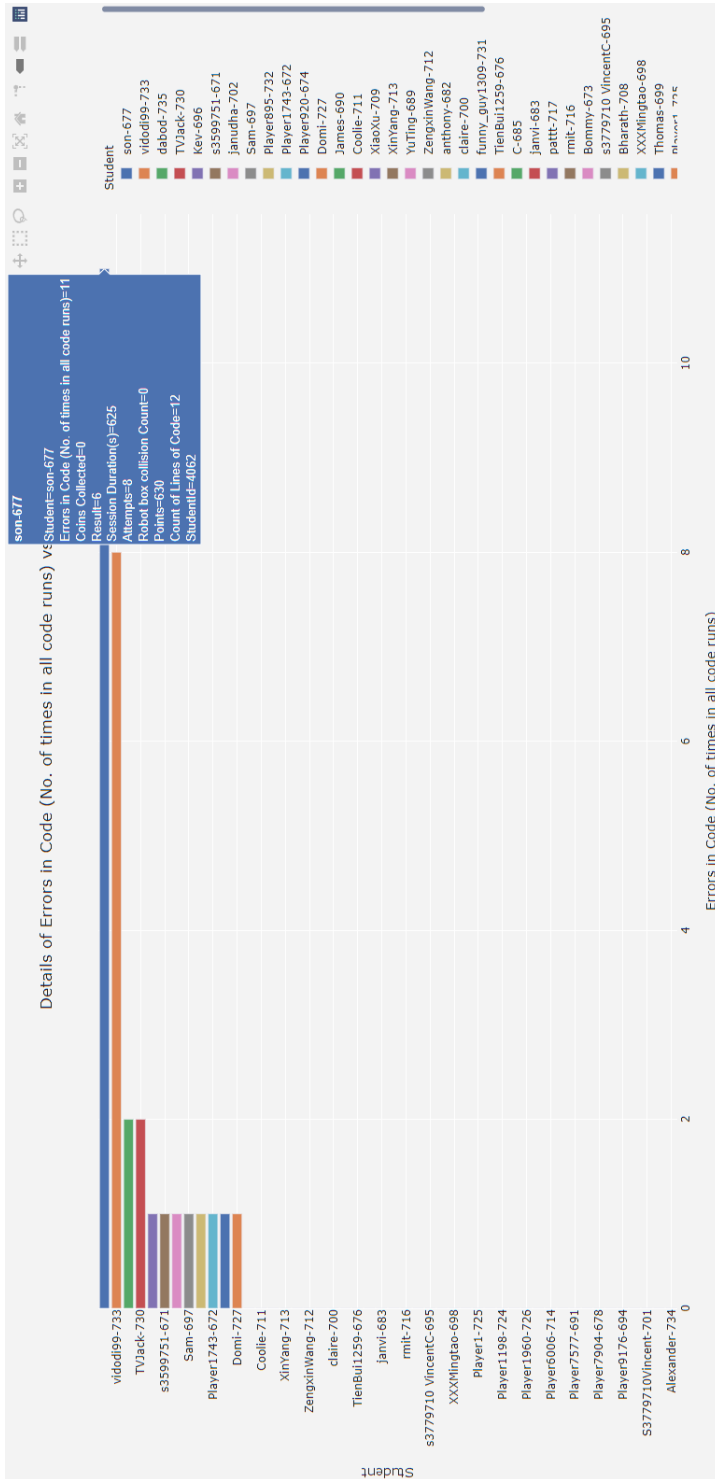


Figure 5.11: A custom plot of the number of errors that occurred in players code submissions

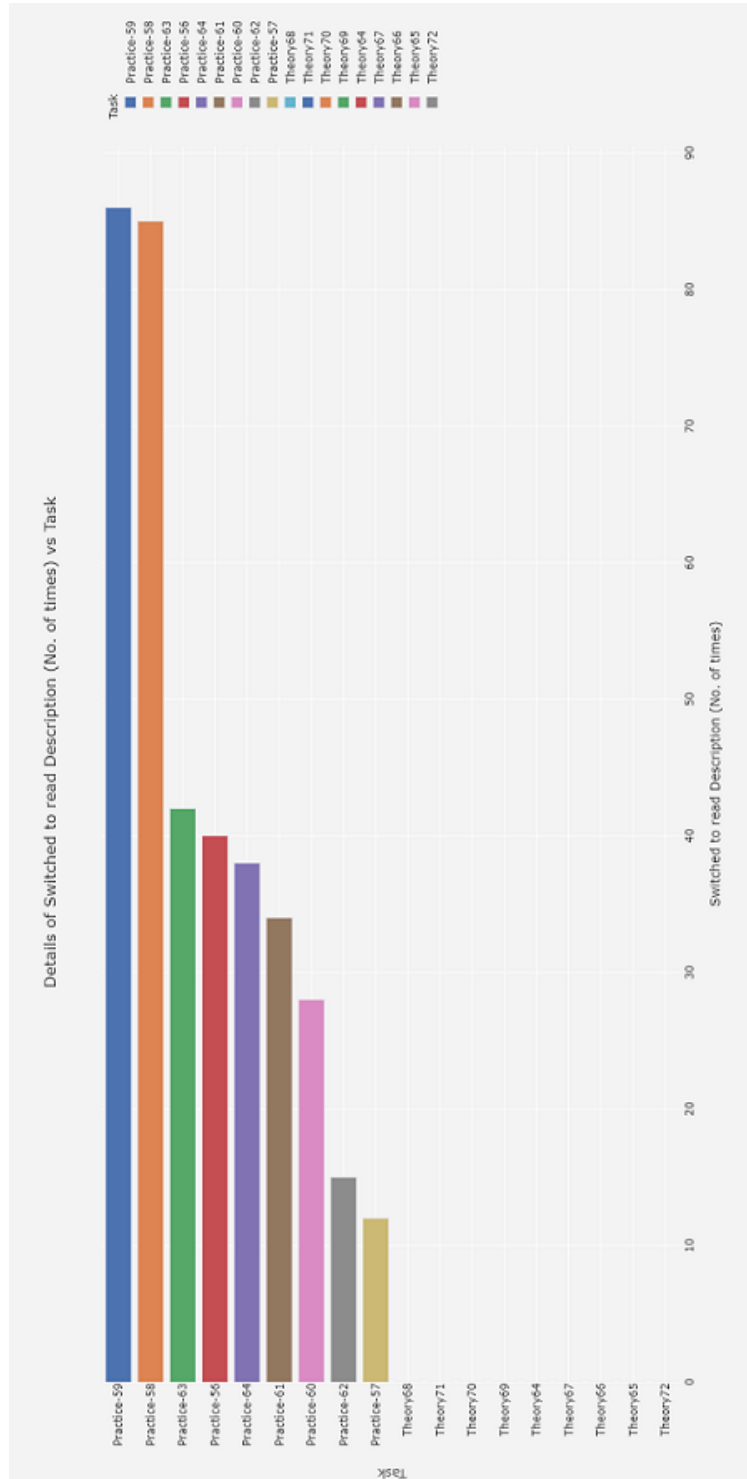


Figure 5.12: A custom plot of the number of times players switched to read description for each of the tasks in a sample class

5.4.2 Administrator User Group

In addition to access to all features available to an educator user, an administrator user can also access complete game data. An administrator can explore data from all the classes/learning activities on sCool and also compare them for further analysis. Figure 5.13 displays the Details section for administrator user.

5.4.2.1 Game Data

This section of the application enables an administrator to explore data of all the learning activities conducted on sCool. An administrator can create simple visualizations consisting of all explorable data. Figure 5.14 displays the game data section with overall game metrics such as number of students on sCool, total amount of time spent by students in the game, points collect, and a custom plot creation form using which administrators can gain additional insights.

5.4.2.2 Learning activities comparison

An administrator can compare learning activities conducted on sCool. An administrator can compare learning activities for various parameters like session duration, points earned, number of items collected. Figure 5.15 displays the learning activity comparison section where users can compare various learning activities. The learning activity with the lowest performance among the selected learning activities is highlighted.

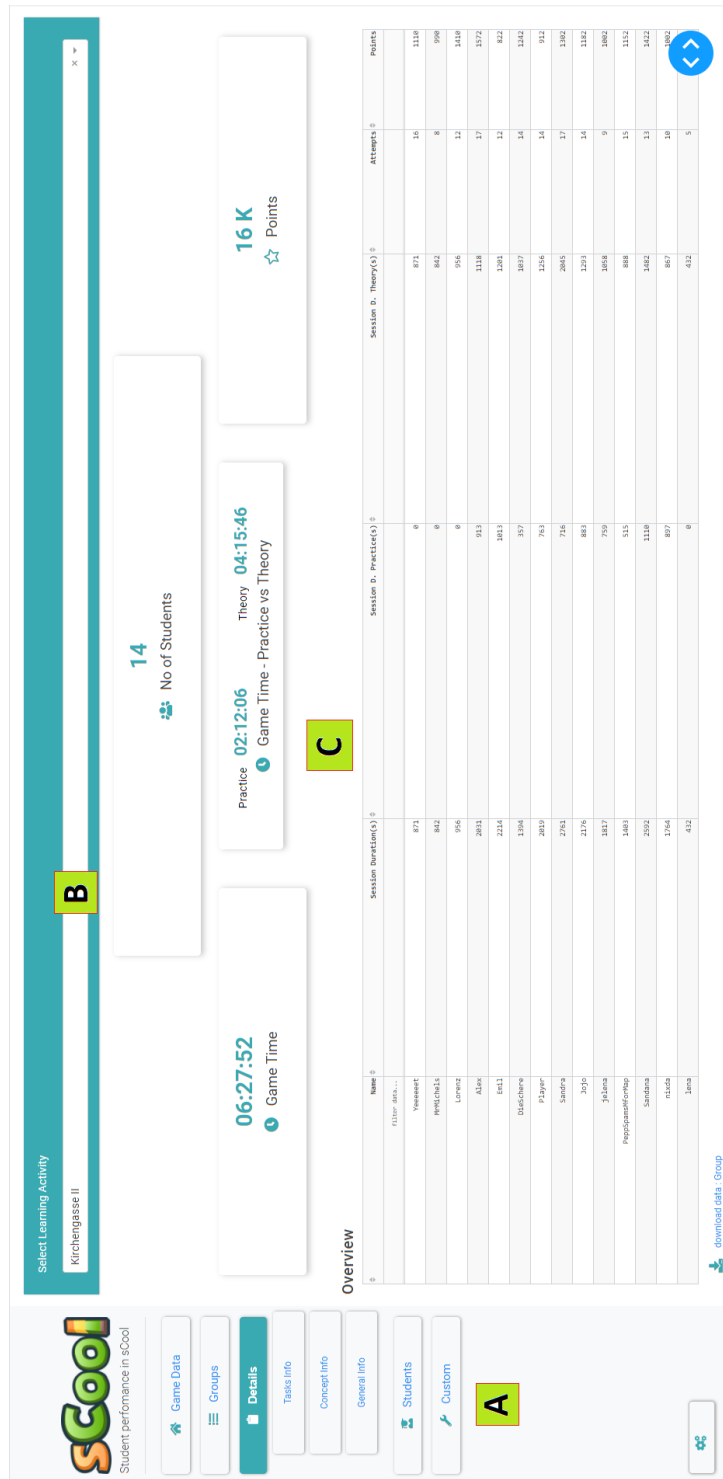


Figure 5.13: Overview of students section

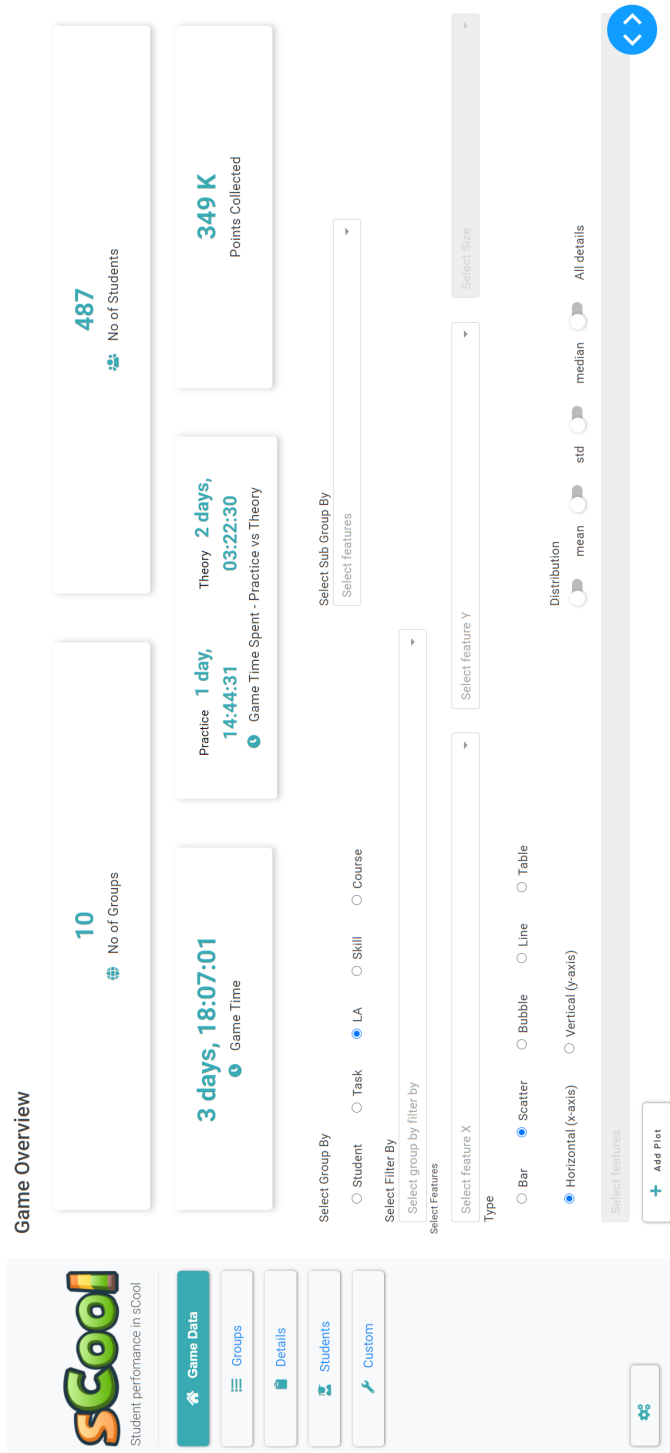


Figure 5.14: Overview of game data section



Figure 5.15: Overview of learning activities comparison section

5.5 Summary

The sCool learning analytics application was developed based on the requirements analysis. Modular implementation was performed to keep the application upgradeable and extendable. The sCool learning analytics application strives to present users with an interactive, informative and straightforward UI. The sCool learning analytics application strives to empower educators and sCool game developers with insights about the student's activities and progress in the courses on sCool game, rather than making a decision tool for them.

The sCool learning analytics application is developed using Plotly's Dash, which is an open-source and full-stack framework. A single sign-on redirect from the sCool web application takes users to the sCool learning analytics application. The sCool learning analytics application connects with the sCool database to gain access to student's game-related data. Data of only those players are analyzed in the application, who had provided explicit permission in the sCool mobile game to permit processing of their game-related data.

The sCool learning analytics application is currently enabled for two stakeholders comprising educators and sCool game developers (administrators), represented by user groups of educators and administrators. Educators can view students' data of their classes/learning activities, whereas administrators (sCool game developers) can view all students' data from all the classes/learning activities on the sCool platform.

The sCool learning analytics application provides its users with details of a class/learning activity with information such as the number of students, an overview of students in the class/learning activity and their scores, tasks completed by students, and coding concepts learned by the students of the class/learning activity. The sCool learning analytics application also provides details of a student with information such as time spent in the game, number of points collected, progress in the course, tasks completed, and game strategies that students undertook to complete the course. In addition to the pre-created information sections, users can also use the custom details section to gain further information such as coding errors by students, syntax error by students, task-wise points earned by students, and skill-wise points

earned by students in the class/learning activity. The sCool learning analytics application provides a sCool game developer or simply administrator with additional features to compare various classes/learning activities and analyze game-related data of all classes/learning activities on the sCool platform.

Chapter 6

Evaluation

In this chapter, evaluation of the learning analytics tool for the coding learning game sCool is described. The first section describes the general scope of the evaluation with an introduction to the research questions. The subsequent sections describe the target groups and participants. The collected information from the evaluation is considered in terms of the research questions.

6.1 Scope

The evaluation consisted of tasks to be performed by participants who try to gain information regarding the performance of a class and involved students, followed by feedback. The participants used the sCool learning analytics application to complete the evaluation. In general, the participants took approximately 10-30 minutes to complete the evaluation.

Three research questions should be answered for the evaluation of the sCool learning analytics application described as:

- **RQ1:** How does the user interface of sCool learning analytics application affect its usability and acceptability?
- **RQ2:** How does sCool learning analytics application assist educators in improving course content based on student's performance?

- **RQ3:** How does sCool learning analytics application assist educators to gain an overview of students performance?

6.2 Instruments and Setup

The evaluation was conducted online since physical interviews are not possible with the COVID-19 pandemic ravaging the entire world. The evaluation has been conducted using *Google Form*¹. Google Form enables users to create online evaluations and analyze results. The evaluation results were downloaded as a *.csv* file and analysed using *Google Sheets*². *Google Sheets* provides functions such as built-in formulas, pivot tables, conditional formatting, with the possibility to create colorful charts and graphs using the data.

An evaluation link was shared with all participants. The participants could complete the evaluation remotely with internet access. The evaluation contained detailed instructions regarding the application and tasks to be performed. Some general questions regarding participant's current profession, education, and familiarity with learning software or e-learning tools were asked in the evaluation. Besides, the participants could provide open-ended feedback about their overall user experience. Additional feedback was collected from the educators regarding whether they would use the application to analyze their students and whether the data collected and analyzed is relevant for educational purposes or for analyzing students' progress.

The sCool learning analytics application was introduced to the participants at the beginning of the evaluation using a pre-recorded video³ introducing the application and its features. The video introduces various features of the application and navigation within the application with textual descriptions. All of the application-specific tasks involved in the evaluation were also briefly described in the video with different examples. It was uploaded on YouTube⁴ and is publicly accessible.

¹<https://www.google.com/forms/about/>

²<https://www.google.com/sheets/about/>

³<https://www.youtube.com/watch?v=atfOw7CoJm0>

⁴<https://www.youtube.com/>

The participants had to complete tasks to find information regarding students of a class using the sCool learning analytics application. The evaluation comprised 11 tasks (numbered T1 to T11) to be completed overall, and the tasks were divided into five subgroups, with each subgroup containing 1-3 questions. Successful completion of each task results in 1 point, whereas failure results in 0 points for the task. A maximum score of 11 could be reached, with a minimum of 0. The tasks were Multiple Choice Question (MCQ), where the participants had to find information using the sCool learning analytics application concerning students of a class and select the correct answers from a list of choices. Table 6.1 provides an overview of the task subgroups and the tasks involved.

All the participants logged in as test users conducting and administrating a class on sCool learning environment. The participants then had to analyze student's performance of their class using sCool learning analytics application. In general, the participants took approximately 10-30 minutes to complete the evaluation.

Table 6.1: Overview of the evaluation tasks and task subgroups

| # | Task | # | Group |
|-----|--|----|---|
| T1 | Which Practice task was completed by least number of students in the entire class? | G1 | Tasks solved by students of the class? |
| T2 | Which Theory task was completed by least number of students in the entire class? | | |
| T3 | How many Theory tasks did student Claire solve? | | |
| T4 | Which of the below students spent least game time (session duration) in the entire class? | G2 | Comparing students of the class |
| T5 | Which student completed least number of Theory tasks in the entire class? | | |
| T6 | claire used which concepts (select all which were used)? | G3 | Coding concepts used in the program submissions |
| T7 | Thomas used which concepts (select all which were used) | | |
| T8 | Which skills did student Alexander complete? | G4 | Finding student's details |
| T9 | Did Thomas complete the course? | | |
| T10 | How many lines of code did Ashley write during the course? | | |
| T11 | To find which student has the most Syntax errors - make a custom plot of 'Syntax Errors in runs (No. of times)' vs 'Student'. Which student has most syntax error in the entire class? | G5 | Plotting a custom plot |

In the evaluation two standardised questionnaires were used: System Usability Scale (SUS) (Brooke, 1995) and NASA⁵ Task Load Index (NASA-TLX) (NASA, 1986; Colligan et al., 2015). The questionnaires are available in the Appendix section.

6.2.1 System Usability Scale (SUS)

System Usability Scale (SUS) has been designed to provide a simple and effective evaluation system for usability measures (Brooke, 1995). The SUS comprises of a ten-item scale providing subjective assessments of usability of the system being evaluated. It is based on *likert scale* (Likert, 1932) where respondents indicate their

⁵<https://www.nasa.gov/>

degree of agreement or disagreement on a 5 point scale with values of *strongly agree*, *agree*, *undecided*, *disagree*, and *strongly disagree*.

The SUS results in a single number representing measure of overall usability of the system being evaluated. In order to calculate the final result below steps can be followed:

- Odd numbered questions: Value of the scores of each of the odd numbered questions are subtracted by 1.
- Even numbered questions: Value of the scores of each of the even numbered questions are subtracted by 5.
- Calculate total score: Sum up the newly calculated values resulting in a total score.
- Calculate final score: Multiply the total score by 2.5 resulting in the final score out of 100.

The scores can be positioned as percentiles for a more meaningful interpretation of the score (Brooke, 1995; A. Bangor et al., 2009; Bangor et al., 2008). A SUS score of 68 is at the center of the range with an average grade C, corresponding to 50th percentile. Table 6.2 provides an overview of the complete curved grading scale with range of SUS score and corresponding grade and percentile range (Lewis and Sauro, 2018).

Table 6.2: Overview of the complete curved grading scale with range of SUS scores and corresponding grade and percentile range (based on Lewis and Sauro, 2018)

| Grade | SUS score | Percentile range |
|--------------|------------------|-------------------------|
| A+ | 84.1 - 100 | 96 - 100 |
| A | 80.8 - 84.0 | 90 - 95 |
| A- | 78.9 - 80.7 | 85 - 89 |
| B+ | 77.2 - 78.8 | 80 - 84 |
| B | 74.1 - 77.1 | 70 - 79 |
| B- | 72.6 - 74.0 | 65 - 69 |
| C+ | 71.1 - 72.5 | 60 - 64 |
| C | 65.0 - 71.0 | 41 - 59 |
| C- | 62.7 - 64.9 | 35 - 40 |
| D | 51.7 - 62.6 | 15 - 34 |
| F | 0 - 51.6 | 0 - 14 |

Although, the SUS does not provide information about the specific issues a system may have, however it provides a measure of usability of the system under consideration. A grade below C could result in consideration of reevaluation of the system's usability as there may be some issues which hinder usability of the system.

6.2.2 NASA Task Load Index (NASA-TLX)

NASA Task Load Index (NASA-TLX) is a subjective assessment tool which helps in evaluation of cognitive impact or perceived workload of a task, or system's performance or effectiveness (NASA, 1986; Colligan et al., 2015; Hart, 2006). In NASA Task Load Index (NASA-TLX) the total workload is divided into six subjective subscales with their description as described below (NASA, 1986):

- **Mental Demand:** How much mental and perceptual activity was required? Was the task easy or demanding, simple or complex?

- Physical Demand: How much physical activity was required? Was the task easy or demanding, slack or strenuous?
- Temporal Demand: How much time pressure did you feel due to the pace at which the tasks or task elements occurred? Was the pace slow or rapid?
- Overall Performance: How successful were you in performing the task? How satisfied were you with your performance?
- Effort: How hard did you have to work (mentally and physically) to accomplish your level of performance?
- Frustration Level: How irritated, stressed, and annoyed versus content, relaxed, and complacent did you feel during the task?

Participants could provide their ratings on a 5 points *likert scale* (Likert, 1932) from 1 to 5 (or 0 to 4) with 1 representing very low to 5 representing very high.

Due to involvement of no physical tasks in the system under consideration and its evaluation, the *Physical Demand* subscale was considered less relevant for the evaluation and hence the subscale in consideration was removed from the evaluation (Hart, 2006).

A downside to NASA-TLX is the more considerable time needed to complete the test and weight calculations. A less demanding version called NASA Raw Task Load Index (NASA-RTLX) could be used (Miller, 2001; Hart, 2006; Moroney et al., 1992). The NASA-RTLX scoring has been found to be almost equivalent/similar to the original TLX scores (Moroney et al., 1992; Miller, 2001). In NASA-RTLX, the score can be computed by taking the sum of the NASA-TLX test ratings and dividing it by six (Miller, 2001), corresponding to the six subscales. In NASA-RTLX, the various components/subscales are not weighed concerning their weights or importance. The NASA-RTLX was used for this study, and the ratings were summed up and divided by 5 (corresponding to the use of only five subscales in this study while excluding the subscale of Physical demand).

6.2.3 Application-related questions

Application-related questions were also asked in every experiment regarding the research question relating to the application. Additional questions were asked to the educators regarding the importance of an analysis tool. The questions were rated with options of *yes*, *maybe*, and *no*. Table 6.3 provides an overview of the various application-related question which were asked to the two subgroups of educators and participants excluding educators.

Table 6.3: Application-related questions

| # | Question | Educators | Others | Type |
|----|--|-----------|--------|--------------|
| Q1 | Would you use the tool for analysis of your students? | ✓ | | yes/maybe/no |
| Q2 | Were you able to identify the progress of a student in the course using the tool? | ✓ | ✓ | yes/maybe/no |
| Q3 | Were you able to identify students who are performing poorly and may need assistance using the tool? | ✓ | ✓ | yes/maybe/no |
| Q4 | Were you able to identify tasks that were easy or hard for students? (For example, the most completed tasks or tasks which no one completed.) | ✓ | ✓ | yes/maybe/no |
| Q5 | Is this tool meaningful? | ✓ | | yes/maybe/no |
| Q6 | Is all the data collected and analyzed relevant for Educational purpose or useful for analyzing students progress? | ✓ | | yes/maybe/no |
| Q7 | Would you recommend this to a colleague? | ✓ | ✓ | yes/maybe/no |

6.3 Participants

The evaluation of the sCool learning analytics application was done with participants from several educational and occupational groups. The evaluation was conducted online due to COVID-19 restrictions. The evaluation was distributed through social media channels. Overall, 22 volunteers participated and completed

the evaluation. 36.4 % (8 participants) of the participants identified their gender as female and the rest 63.6 % (14 participants) as male. Figure 6.1 shows an overview of the gender of participants. Figure 6.2 shows an overview of the age of participants. The participant's ages ranged between 26-35 years. The average age was 29.64 years (Median=29 years, SD=3.11).

The participants included students and researchers at TU Graz, software developers, data analysts, and school teachers in Vienna, India and Vietnam. 18.2 % (4 participants) of the participants were professional teachers/educators, 54.5 % (12 participants) of the participants were expert professional users who have a master's or higher degree in Computer Science or closely related fields, 9.1 % (2 participants) of the participants were current students, and 18.2 % (4 participants) were in other fields of employment. Figure 6.3 depicts an overview of the profession of the participants. The participants could be classified into two main subgroups of educators and participants excluding educators based on their occupation as the two subgroups members could have varying levels of experiences in using educational tools to analyze students' performance. The results of the evaluation are considered for these subgroups in subsequent sections.

Figure 6.4 shows an overview of the highest educational degree of participants depicting that 9.1 % (2 participants) of the users already had a Doctorate, 72.7 % (16 participants) of the users had a Master's degree, and 18.2 % (4 participants) had a Bachelor's degree. All the educators had a Master's degree. Figure 6.5 shows an overview of familiarity of participants with e-learning tools or learning software, depicting that the majority of the participants, 68.2 % (15 participants), were familiar with an e-learning tool. 27.3 % (6 participants) of the participants were somewhat familiar, and 4.45 % (1 participant) of the participants responded with no familiarity with an e-learning tool or learning software.

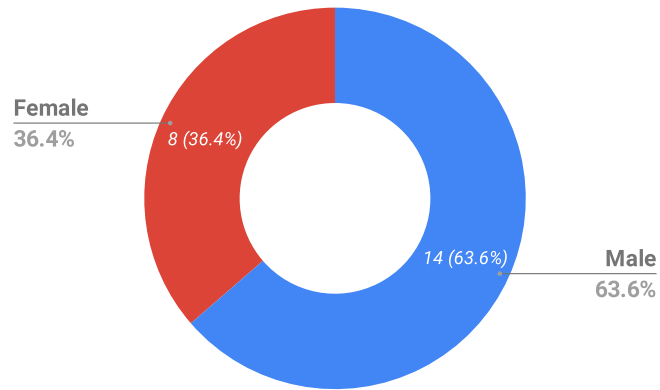


Figure 6.1: Gender of participants

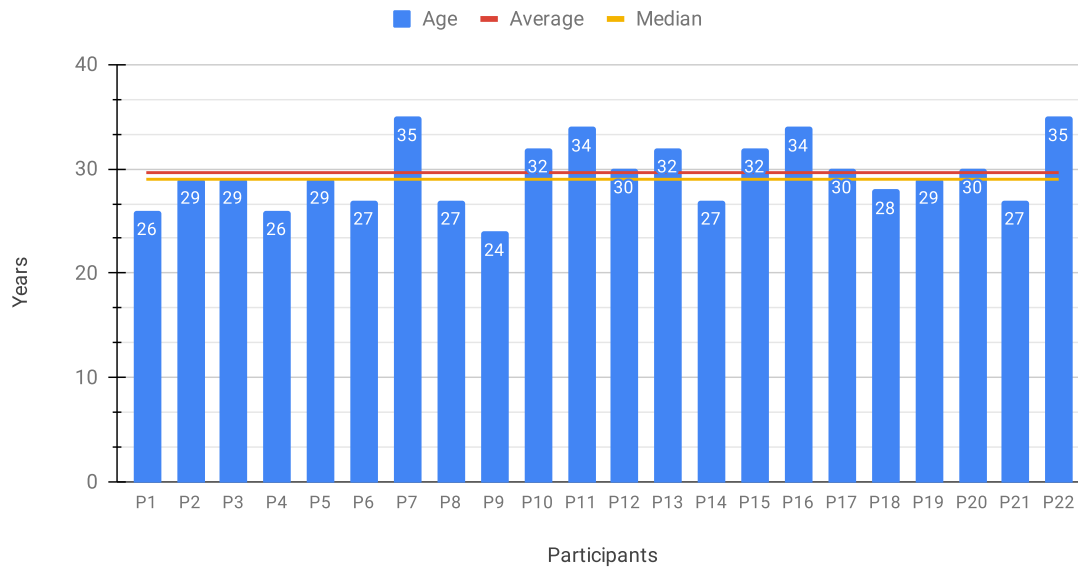


Figure 6.2: Age of participants (in years)

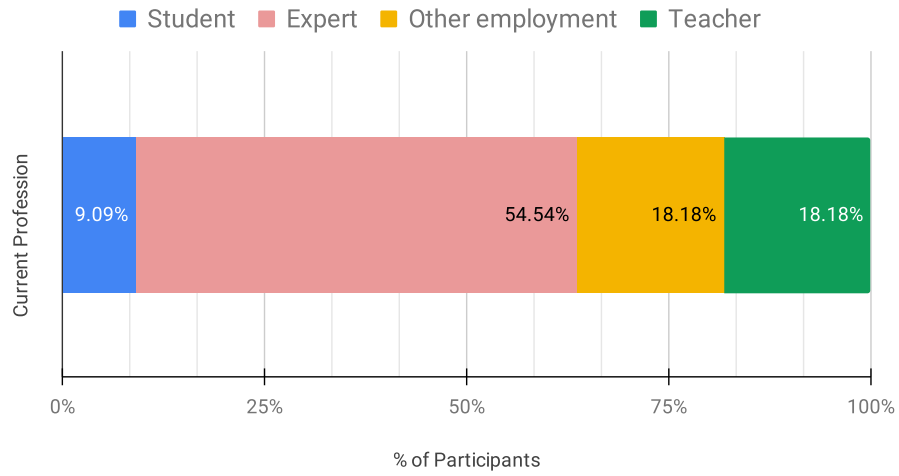


Figure 6.3: Profession of participants

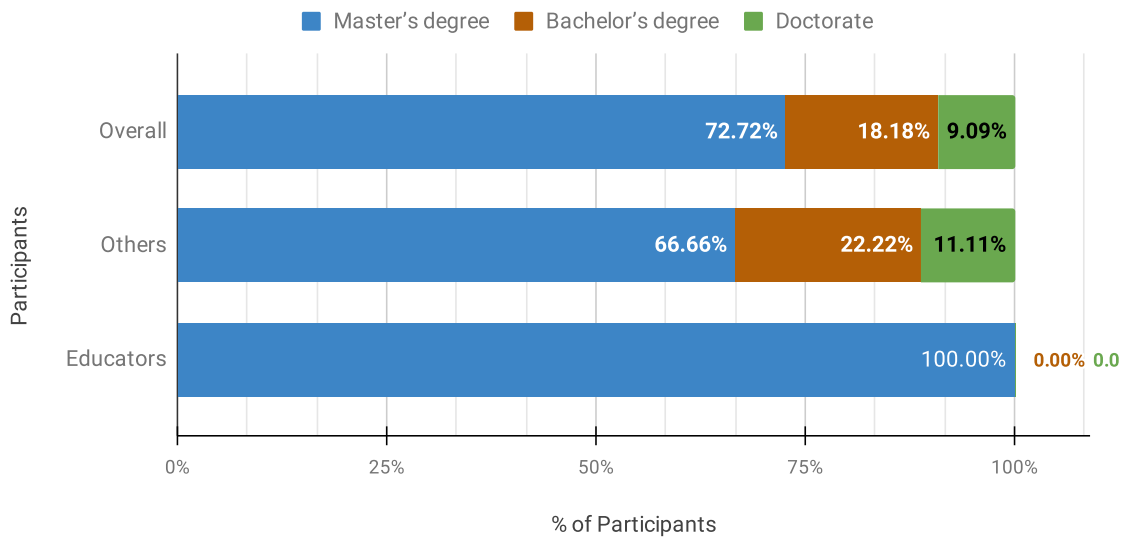


Figure 6.4: Highest educational degree of participants

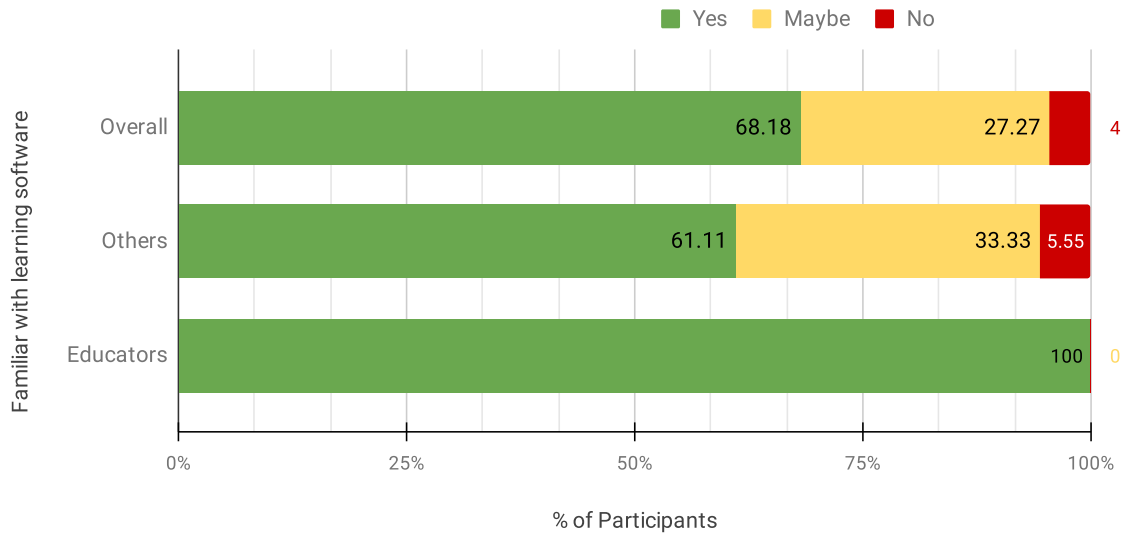


Figure 6.5: Familiarity of participants with e-learning tools or learning software

6.4 Evaluation and Results

This section discusses the evaluation results as 22 participants participated and completed the evaluation and provided their feedback. The following subsections describe the completion of the tasks by the participants and their scores. The subsections also provide the evaluation results and feedback from the participants.

6.4.1 Task Completion and Scores

Figure 6.6 shows an overview of the distribution of evaluation scores of results of tasks performed by participants during the evaluation. The evaluation comprised 11 tasks to be completed where successful completion of a task results in 1 point, whereas failure results in 0 points for the task. A maximum score of 11 could be reached, with a minimum of 0. The average score of participants was 9.68 points (Median=10 points, SD=1.36) from the maximum possible score of 11 points, with 36.36 % (8 participants) participants recording 11 points. A minimum score of 6 was recorded

by 4.5 % (1 participant) of the participants. Figure 6.7 shows an overview of evaluation scores of various participant occupational groups (Educators and other participants). The mean score of educators subgroup was 9.75 (Median=9.5, SD=0.96) and the mean scores of other participants (excluding educators) was 9.67 (Median=10, SD=1.46). The evaluation scores show that both participant subgroups could use the application and gain information at similar levels despite varying occupational experiences in using educational tools to analyze students' performance.

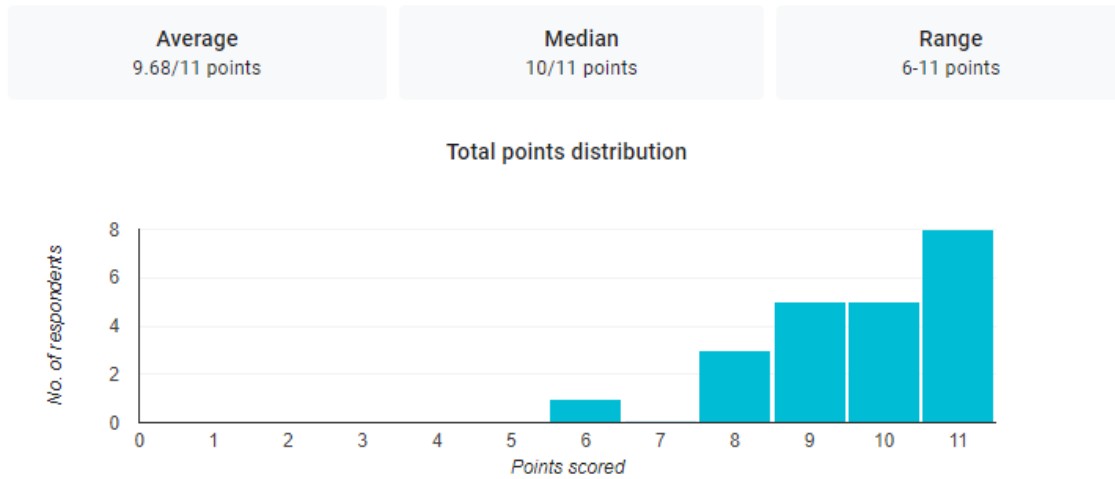


Figure 6.6: Distribution of evaluation scores of all participants for task completion

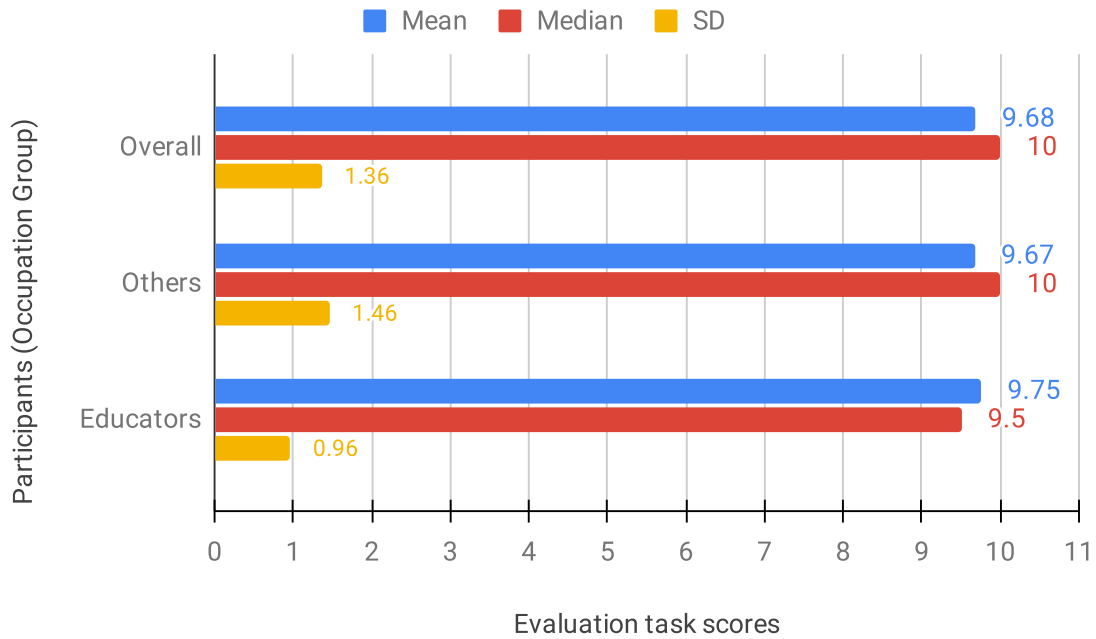


Figure 6.7: Evaluation scores of participants for task completion

Table 6.4 provides an overview of the various tasks (numbered T1 to T11) and the success and fail percentage in the respective tasks. The task (T10) to find the number of lines of code submitted by a student was successfully completed by all of the attempting participants. The participants were also able to identify that the student under consideration submitted zero lines of code, implying that the student had not submitted any programming solutions in the entire course. The majority of the participants, 81.81 % (18 participants), were able to identify the student’s course progress (T9). All the participants were able to identify the number of tasks completed by a student under consideration (T3). All the participants successfully identified the student who spent the least amount of time completing the course (T4). A majority, 86.36 % (19 participants), of the participants, were able to identify the student who could complete only the least number of tasks (T5). Such critical information has provided participants key insights about students who may be lacking course progress by either being unable to write some lines of code,

or unable to make sufficient progress in the course. Concerning **RQ3**, the insights mentioned above could help educators gain an overview of students of their class and even identify students who are performing poorly.

Almost all of the participants were able to identify the tasks (T1 and T2) completed by the least number of students in both concept-learning (100 %; 22 participants) and practical (95.45 %; 21 participants) sections of the game, respectively. All of the participants who attempted the tasks (T6 and T7) were able to complete them and identify the programming concepts learned by students to solve the coding course tasks on sCool. 95.45 % (21 participants) of the participants were also able to identify the skills acquired by the student under consideration (T8). The information gathered could provide educators and game developers key insights regarding course content. The course tasks which students are unable to complete may either be too difficult for them, not engaging to them, too easy to keep them motivated, or unreachable in the course levels. Some of the tasks could also act as stumbling points for students, which hinders their course progress. It is critically important to be able to identify such stumbling tasks. The educators and game developers could further utilize the newfound information to improve the course content further or in creating new course content (**RQ2**). Similarly, when students have completed all the course levels, the educators can further create course content with increased difficulty. Additionally, educators and game developers can also identify individual student's progress and create additional course content for fast learners with increased difficulty levels (**RQ2**).

The majority of the tasks were completed successfully by the participants. The task (T10) to find the number of lines of code submitted by student Ashley was successfully completed by all participants who attempted it, 63.63% (14 participants), however 36.36% (8 participants) of the participants skipped the task or missed to submit their answers. However, the task (T11) to find the student with the most syntax errors was successfully completed by 55% (11 participants) of the participants who attempted it (overall 50% of the participants; 11 participants).

Table 6.4: Overview of the task wise answers

| Group | # | Task | Success | Fail | Missed |
|-------|-----|--|---|--------------------------------------|------------|
| G1 | T1 | Which Practice task was completed by least number of students in the entire class? | 95.45% (21) | 4.45% (1) | 0 |
| | T2 | Which Theory task was completed by least number of students in the entire class? | 100% (22) | 0 | 0 |
| | T3 | How many Theory tasks did student Claire solve | 100% (22) | 0 | 0 |
| G2 | T4 | Which of the below students spent least game time (session duration) in the entire class | 100% (22) | 0 | 0 |
| | T5 | Which student completed least number of Theory tasks in the entire class | 86.36% (19) | 13.63% (3) | |
| G3 | T6 | claire used which concepts (select all which were used) | 100% of attempted (95.45% overall) (21) | 0 | 4.45% (1) |
| | T7 | Thomas used which concepts (select all which were used) | 100% (22) | 0 | 0 |
| G4 | T8 | Which skills did student Alexander complete | 95.45% (21) | 4.45% (1) | 0 |
| | T9 | Did Thomas complete the course Coding-2020? | 81.81% (18) | 18.18% (4) | 0 |
| | T10 | How many lines of code did Ashley write during the course? | 100% of attempted (63.63% overall) (14) | 0 | 36.36% (8) |
| G5 | T11 | To find which student has the most Syntax errors - make a custom plot of 'Syntax Errors in runs (No. of times)' vs 'Student'. Which student has most syntax error in the entire class? | 55% of attempted (50% overall) (11) | 45% of attempted (40.9% overall) (9) | 9.09% (2) |

6.4.2 Evaluation Responses of Participants for Application Specific Question

The participants could be classified into two main subgroups of educators and participants excluding educators. The following subsections provides the evaluation responses of participants of the two subgroups.

6.4.2.1 Educator Participants

Figure 6.8 shows an overview of results of the evaluation by participants who are educators who responded that 100 % of them were able to identify the progress of a student in the course using the sCool learning analytics application (Q2). Figure 6.9 shows that all of the participating educators (100 %; 4 participants) were able to identify students who are performing poorly and may require assistance (Q3) based on various game metrics such as number of tasks completed, number of lines of code written, and points collected using the sCool learning analytics application (**RQ3**). Figure 6.10 shows that most participating educators (75 %; 3 participants) were able to identify easy or complex tasks for students (Q4) as the most completed or least completed tasks (**RQ2**). In contrast, 25 % (1 participant) of the participant educators reported that they were unable to identify easy or complex tasks for students (Q4). Table 6.5 provides an overview of response to application-specific question by the participants who provided responses to the questions.

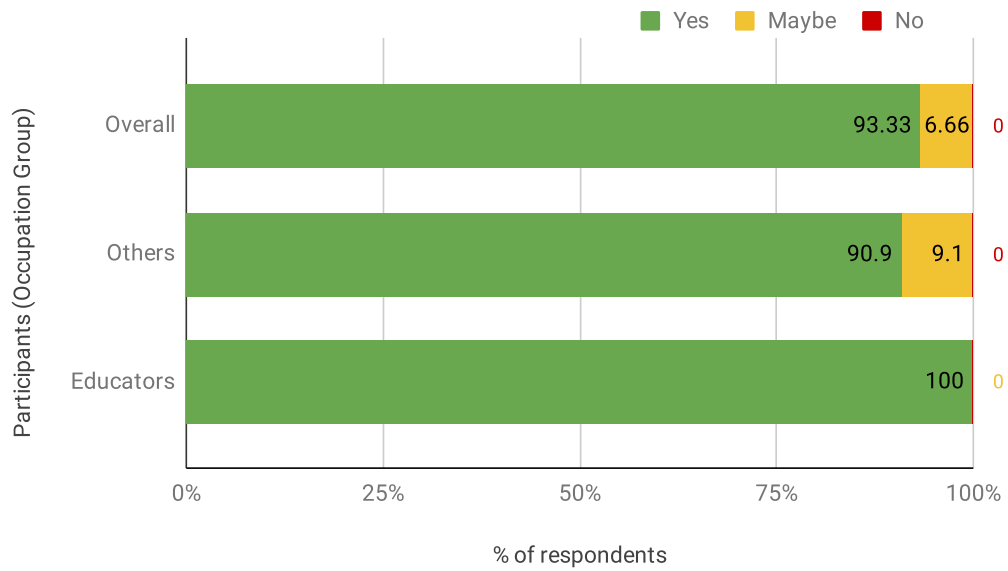


Figure 6.8: Able to identify course progress of students

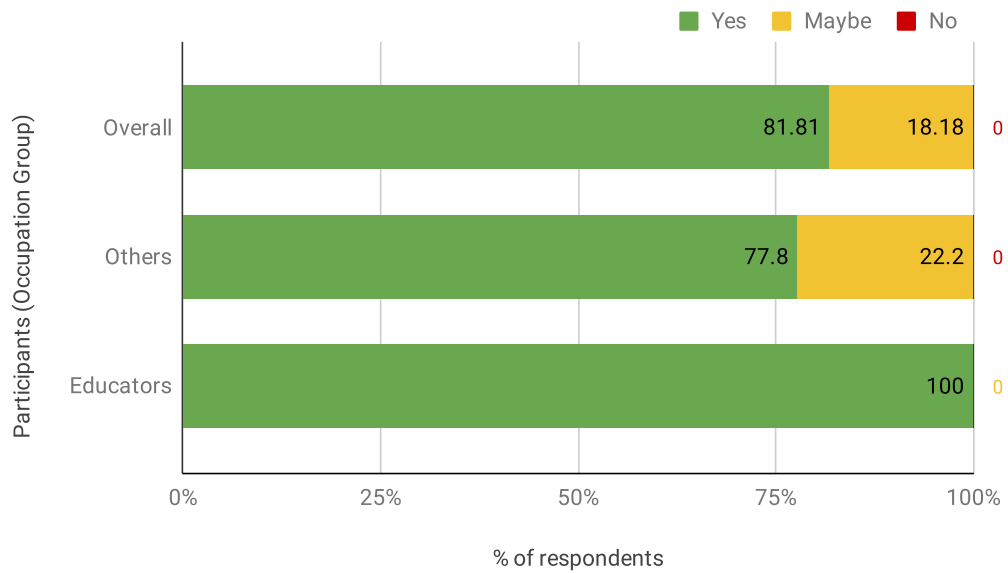


Figure 6.9: Able to identify students performing poorly

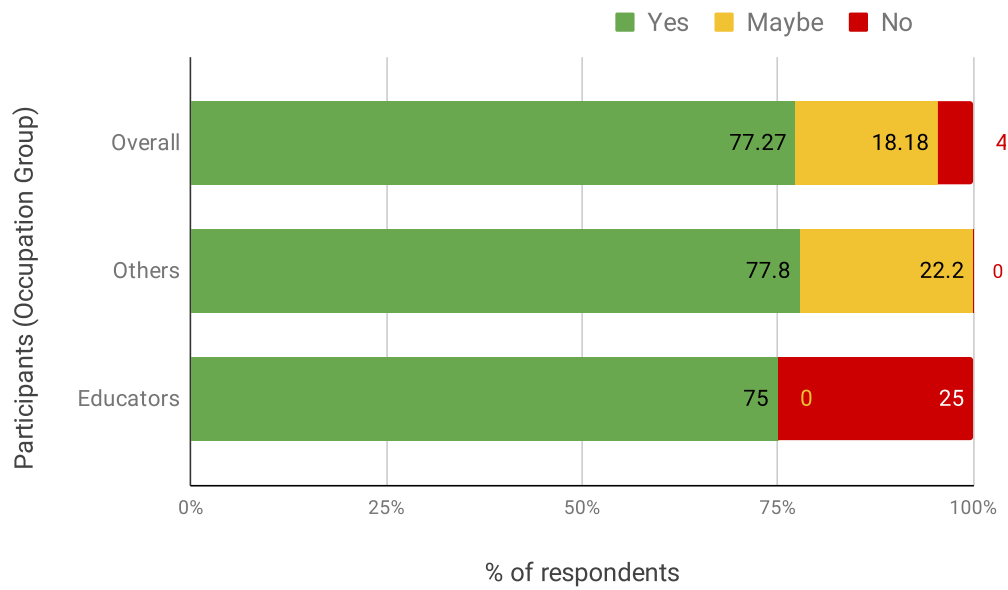


Figure 6.10: Able to identify tasks which were hard or easy

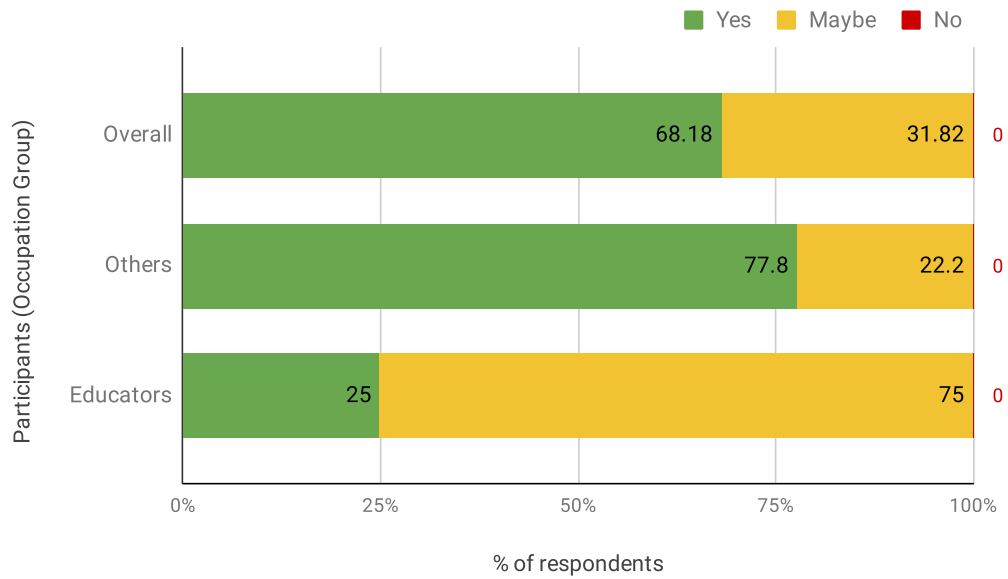


Figure 6.11: Would recommend application to a colleague

Table 6.5: Overview of the responses of application-specific questions

| | Response | Overall | Others | Educators |
|--|-----------------|-----------------|----------------|------------------|
| Were you able to identify the progress of a student in the course using the tool? | Yes | 93.33 % (14) | 90.9 % (10) | 100 % (4) |
| | Maybe | 6.66 % (1) | 9.1 (1) | 0 |
| | No | 0 | 0 | 0 |
| Were you able to identify students who are performing poorly and may need assistance using the tool? | Yes | 81.81 % (18) | 77.8 % (14) | 100 % (4) |
| | Maybe | 18.18 % (4) | 22.2 % (4) | 0 |
| | No | 0 | 0 | 0 |
| Were you able to identify tasks that were easy or hard for students? (For example, the most completed tasks or tasks which no one completed.) | Yes | 77.27 % (17) | 77.8 % (14) | 75 % (3) |
| | Maybe | 18.18 % (4) | 22.2 % (4) | 0 |
| | No | 4.5 % (1) | 0 | 25 % (1) |
| Would you recommend application to a colleague? | Yes | 68.18 % (15) | 77.8 % (14) | 25 % (1) |
| | Maybe | 31.82 % (7) | 22.2 % (4) | 75 % (3) |
| | No | 0 | 0 | 0 |

Figure 6.12 shows that the individual perspective of the application being meaningful (Q5) was considered positively by half (50 %; 2 participants) of the participating educators. At the same time, half (50 %; 2 participants) of the participating educators remained undecided about the application being meaningful and responded with *maybe*. Figure 6.13 shows that half (50 %; 2 participants) of the participating educators agreed that the data collected and analyzed is relevant for educational purposes or analyzing student’s progress in a class (Q6). In comparison, the other half (50 %; 2 participants) of the participating educators remained undecided regarding the data collected for educational purposes or analyzing students’ progress in a class and responded with maybe. Figure 6.14 shows that the majority (50 %; 2 participants) of the educators remained undecided about using the application for analyzing students (Q1), and amongst the remaining (25 %; 1 participant) would use the application for analyzing students and remaining (25 %; 1 participant) could not find the use of the information provided by the application. This also provides information that the participants may be unclear about using information from players game-related data for analyzing students.

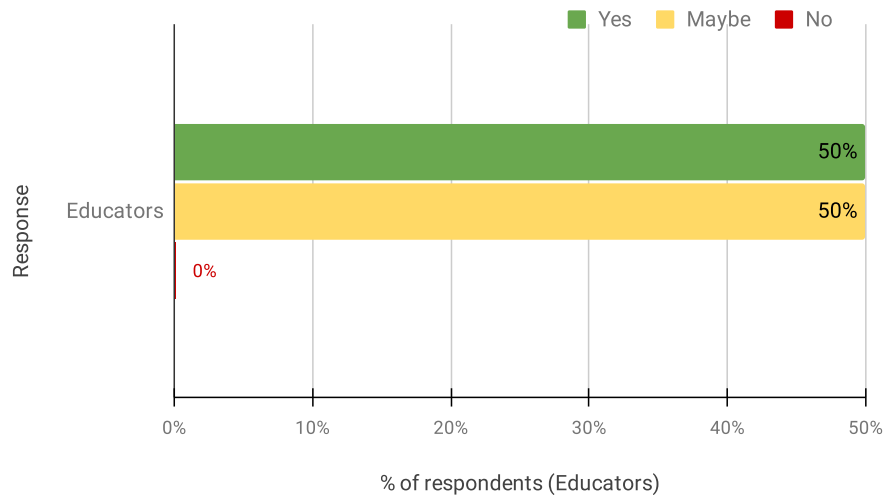


Figure 6.12: (Educators) Found the tool meaningful

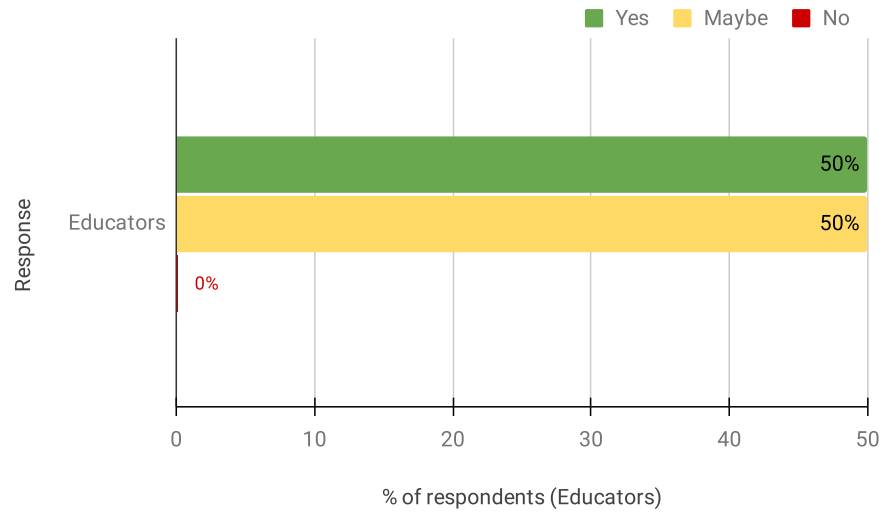


Figure 6.13: (Educators) Found the data collected and analyzed relevant for Education purpose or useful for analyzing students progress

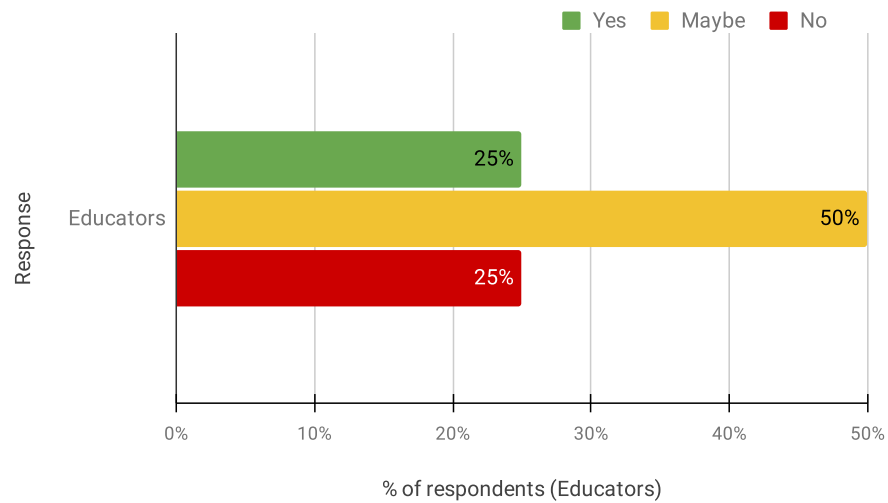


Figure 6.14: (Educators) Would like to use the application for analyzing students

6.4.2.2 Participants Excluding Educators

This subsection presents the evaluation results submitted by participants, excluding educators. Figure 6.8 shows that majority, 90.9 % (10 participants), of the respondents who provided feedback for this question agreed that they were able to identify student's progress in the course (Q2) using the sCool learning analytics application (**RQ3**). Figure 6.9 shows that majority, 77.8 % (14 participants), of the respondents agreed that they were able to identify students who are performing poorly and may require assistance using the sCool learning analytics application (Q3) based on game metrics such as number of tasks completed, number of lines of code submitted, and points collected (**RQ3**). 22.2 % (4 participants) of the respondents responded with a maybe and remained undecided. Figure 6.10 shows that majority (77.8 %; 14 participants) of the respondents, agreed that they were able to identify tasks that were easy or hard for students (Q4) using the sCool learning analytics application based on metrics such as most completed tasks or tasks which no student could complete (**RQ2**). 22.2 % (4 participants) of the respondents responded with a maybe and remained undecided. Figure 6.11 shows that the majority (77.8 %; 14 participants) agreed that they recommend the sCool learning analytics application to their colleagues (Q5), and the positive recommendation feedback could be interpreted as acceptability concerning **RQ1**. 22.2 % (4 participants) of the respondents responded with a maybe and remained undecided.

6.4.3 System Usability Scale (SUS)

Figure 6.15 shows an overview of the results of System Usability Scale (SUS) scores of each of the participants. The average SUS score is 81.70 (Median=85, SD=11.32), ranging from 52.5-97.5, with 52.5 being recorded as the lowest score and 97.5 as the highest. A SUS score of 81.70 implies that the overall application satisfies the usability criteria. Concerning **RQ1**, from the SUS score results, it can be observed that the user interface of the sCool learning analytics application satisfies usability criteria.

SUS scores of participants

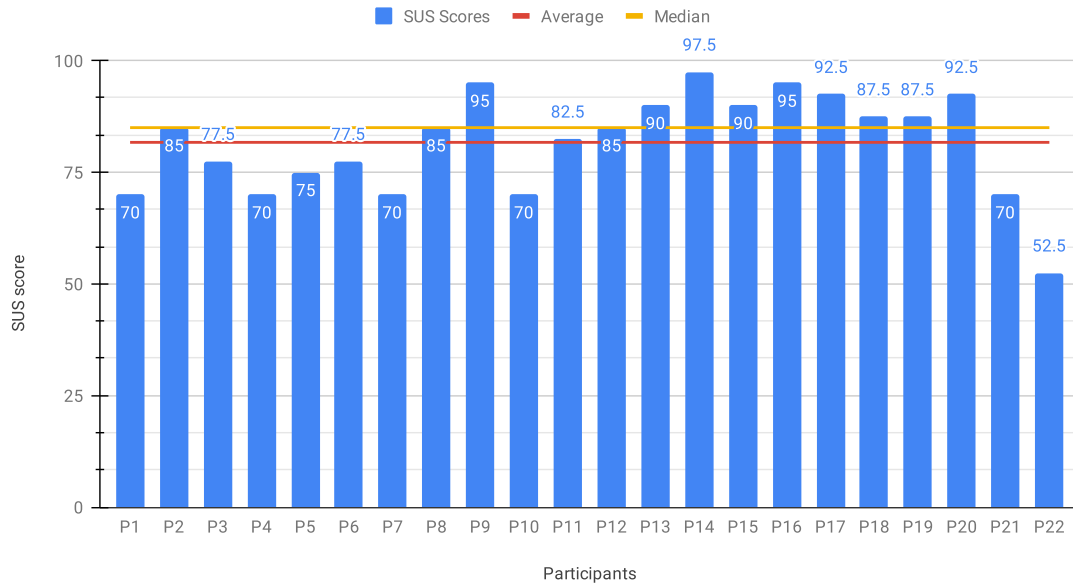


Figure 6.15: Results of SUS scores of each participants in the evaluation

Table 6.6 provides an overview of SUS scores per questions of the SUS scale. It could be seen that the participants rated the application as easy to use. However, 9.09 % (2 participants) of the participants responded that they would not like to use the application frequently with a strong disagreement, 13.63 % (3 participants) responded that they would like to use the application frequently with a strong agreement, and 45.45 % (10 participants) responded with an agreement.

Most of the participants strongly disagreed (31.81 %; 7 participants) or disagreed (59.09%; 13 participants) that the application was unnecessary complex. 9.09 % of the participants found the application unnecessarily complex, with agreement (4.45 %; 1 participant) and strong agreement (4.45 %; 1 participant) equally amongst them. Most of the participants strongly agreed (40.09 %; 9 participants) or agreed (40.09 %; 9 participants) that the application was easy to use. The remaining participants were undecided (9.09 %; 2 participants) or disagreed (9.09 %; 2 participants) with the statement that the application was easy to use.

The majority of the participants (54.54 %; 12 participants) strongly disagreed that they need the help of a technical person to use the application. The remaining participants either disagreed (27.27 %; 6 participants) or remained undecided (18.18 %; 4 participants). The majority of the participants (72.72 %; 16 participants) agreed to have found the various functions in the application to be well integrated, and 13.63 % (3 participants) strongly agreed to the same.

The majority (63.63 %; 14 participants) of the participants responded that they strongly disagreed that there was too much inconsistency in the application. Among the remaining, 22.72 % (5 participants) were in disagreement, and 9.09 % (2) remained undecided. However, 4.45 % (1 participant) of the participants strongly agreed that there was inconsistency in the application. Most of the users were in strong agreement (36.36 %; 8 participants) and agreement (45.45 %; 10 participants) with the statement that most people would learn to use this application very quickly. The remaining 18.18 % (4 participants) users remained undecided. The majority (81.81 %; 18 participants) of the participants did not find the application awkward to use. Most of the participants responded that they felt confident using the application with a strong agreement (45.45 %; 10 participants) and an agreement (40.09 %; 9 participants) , respectively. The majority (86.36 %; 19 participants) of the participants responded that they did not need to learn many things before they could get going with the application.

One of the participants reported a SUS score of 52.5, which is below the average SUS score (refer table 6.2), implying usability issues for first-time users. Future work could therefore be an onboarding tutorial for new users.

Table 6.6: Overview of the responses on SUS

| | Strongly Disagree | Disagree | Undecided | Agree | Strongly Agree |
|---|--------------------------|-----------------|------------------|--------------|-----------------------|
| I think I would like to use this application frequently | 9.09% (2) | 4.45% (1) | 27.27% (6) | 45.45% (10) | 13.63% (3) |
| I think I found the application unnecessary complex | 31.81% (7) | 59.09% (13) | 0 | 4.45% (1) | 4.45% (1) |
| I thought this application was easy to use | 0 | 9.09% (2) | 9.09% (2) | 40.09% (9) | 40.09% (9) |
| I think that I would need the help of a technical person to use this application | 54.54% (12) | 27.27% (6) | 18.18% (4) | 0 | 0 |
| I found the various functions in this application were well integrated | 0 | 0 | 13.63% (3) | 72.72% (16) | 13.63% (3) |
| I thought there was too much inconsistency in the application | 63.63% (14) | 22.72% (5) | 9.09% (2) | 0 | 4.45% (1) |
| I would imagine that most people would learn to use this application very quickly | 0 | 0 | 18.18% (4) | 45.45% (10) | 36.36% (8) |
| I found this application very awkward to use | 81.81% (18) | 13.63% (3) | 4.45% (1) | 0 | 0 |
| I felt very confident using this application | 0 | 0 | 13.63% (3) | 40.09% (9) | 45.45% (10) |
| I needed to learn a lot of things before I could get going with this application | 86.36% (19) | 9.09% (2) | 4.45% (1) | 0 | 0 |

6.4.4 NASA-TLX

Table 6.7 provides the NASA-RTLX rating results and scores concerning perceived workload regarding solving the tasks within the task subgroups. Participants provided their perceived workload ratings on 5 points *likert scale* (Likert, 1932) from 1 to 5 with one representing very low to five representing very high. From these results, it can be observed that the tasks about creating a custom plot (G5) have the highest perceived workload (Mean=2.51, Median=2.3, SD=0.75) among the tasks as the participants stated feelings of higher mental activity, temporal demand, effort, and frustration level compared to other tasks subgroups while reporting lower performance. This indicates that tasks related to creating a custom plot require a (slightly) higher workload compared to performing other tasks using the application. The task subgroup comparing students of the class (G2) has the second-highest workload (Mean = 2.49, Median=2.4, SD=0.56) from participant ratings. The task subgroup about identifying coding concepts used in the program submissions by students (G3) has the lowest mental demand (Mean = 2.24, Median=2, SD=0.57) among the task subgroups. Overall, creating custom plots (G5) and comparing students of a class (G2) require a higher workload. From the RTLX scores, it can be observed that none of the tasks were perceived as tasks requiring an intense workload, nearing scores of 5 (on a scale of 1 to 5), with a maximum rating of 2.51 (Median=2.3, SD=0.75) only (**RQ1**).

Table 6.7: NASA-RTLX results

| Group | Tasks | Mental Demand | Temporal Demand | Overall Performance | Effort | Frustration Level | RTLX scores |
|-------|------------|---|---------------------------------------|---|---|---|---|
| G1 | T1, T2, T3 | Mean = 1.64 (Median=1.5, SD=0.73) | Mean = 2 (Median=2, SD=0.98) | Mean = 4.32 (Median=4.5, SD=0.95) | Mean = 1.82 (Median=2, SD=0.91) | Mean = 1.5 (Median=1, SD=0.86) | Mean = 2.26 (Median=2.1, SD=0.54) |
| G2 | T4, T5 | Mean = 2.05 (Median=2, SD=0.90) | Mean = 2 (Median=2, SD=0.76) | Mean = 4.27 (Median=4.5, SD=0.98) | Mean = 2.23 (Median=2, SD=0.75) | Mean = 1.86 (Median=1.5, SD=1.13) | Mean = 2.49 (Median=2.4, SD=0.56) |
| G3 | T6, T7, T8 | Mean = 1.54 (Median=1, SD=0.96) | Mean = 1.68 (Median=1, SD=0.95) | Mean = 4.5 (Median=5, SD=0.96) | Mean = 1.72 (Median=1.5, SD=0.88) | Mean = 1.73 (Median=1.5, SD=0.83) | Mean = 2.24 (Median=2, SD=0.57) |
| G4 | T9, T10 | Mean = 1.95 (Median=2, SD=1) | Mean = 1.91 (Median=2, SD=1.02) | Mean = 4.23 (Median=4, SD=0.97) | Mean = 1.91 (Median=2, SD=1.02) | Mean = 1.77 (Median=1.5, SD=0.87) | Mean = 2.36 (Median=2.3, SD=0.66) |
| G5 | T11 | Mean = 2.38 (Median=2, SD=1.12) | Mean = 2.05 (Median=2, SD=1.36) | Mean = 4.05 (Median=4, SD=1.2) | Mean = 2.24 (Median=2, SD=1.34) | Mean = 1.9 (Median=2, SD=1.26) | Mean = 2.51 (Median=2.3, SD=0.75) |

6.4.5 Suggestions and Feedback

Some of the participants also provided suggestions and feedback to improve the application. In general, participants suggested providing a help feature for beginners to shorten the learning curve. The participants also provided suggestions to improve the usability further.

6.4.5.1 Educator Participants

The participants who were educators responded positively and also provided suggestions for improvement with open-text feedback. One of the participants who were educators showed interest in the sCool project and would like to learn more about sCool. The participant would also like to implement and use sCool for teaching students of their school. An additional suggestion was to improve the interface by improving its visual appeal and providing information about the purpose of the tabs. Two of the participants responded with textual feedback as:

- *"Im interested in learning more about the project and potentially get information"*
- *"I think the interface can be made better, more visually appealing, the tabs can probably give some information about their purpose. Overall brilliant visualization and analysis from a teachers/ organizational point of view"*

6.4.5.2 Participants Excluding Educators

The participants, excluding educators, also provided positive feedback and suggestions for improving the application and user interface. In general, the participants provided a suggestion for implementing a help feature for beginners. Since the application could be overwhelming for a new user, the participants provided suggestions also to minimize the displayed data points or game metrics for a quick overview of students. The participants also suggested improving the filter options in the data table. The participants also emphasized that currently, the custom plots printed

only every second name of a student due to less screen space available or many students, which confuses the users. The participants also suggested providing a message informing users that they must select a class(learning activity) to see details of the selected class. An additional suggestion was to highlight the tasks with a high and low success and fail number to provide quick information. The participants also suggested reducing the evaluation itself since it seemed very long. Some of the textual feedback are listed below as:

- *"some minor UX issues but it is still useful for visualizing the data."*
- *"Great!"*
- *"The system seems user friendly and good to use, but please improve this survey it is really long xD"*
- *"All panels in a single page, confusing to view graphs. Also, when each student is selected the font needs to highlight the important features. After selecting a student name and want to try another student cursor didn't move to right most position. Had to just click on delete with cursor before first character."*
- *"As an end user, The number of variables that a teacher has to skim through per student, in order to understand where the student's strengths and weakness shall be minimized. If the app displayed these minimum set of variables for the student, and in the end can also have a graph or something like that which would inform the teacher how well the student performed in different tasks/-concepts/games.. and also, amongst the peers all in a single page.. this would help minimize the amount of time the teacher has to spend navigating through the app to study about the student performance."*
- *"Under Details->Overview->Show overview data: It would be useful if the filtering in the table worked better, e.g. support wild cards, match also entries that don't match case, etc."*
- *"I really like the overall concept and the flexibility of the tool. It gives a lot of insight from different perspectives and would be very useful in a class setting for"*

identifying what concepts students have issues with and which students might need some additional help. I especially loved that I can create custom charts based on custom features I select. The only aspect which needs improvement is the overall user experience. The tool is currently too overwhelming when you start using it. Maybe adding some UI guide and also further segmenting the tasks into sub groups / tabs would lower the initial mental burden for a user."

- *" -) Custom plot features in drop-down not alphabetical! -) Maybe mark with colors which tasks have a high and a low success/fail ratio -) The expand buttons for the plots are not very well separated from the background. Make buttons instead of just clickable strings."*
- *"In a plot, I think it is pointless to print every second name beside the bar, especially if you have a key on the side of the chart. Either you print them all or none of them. In the 'Students' tab, when the learning activity is not selected, you cannot search for a student. This is of course right, but a message informing me that I should first select a learning activity could clear doubts in certain situations."*

6.5 Discussion and Limitations

Based on the feedback from the participants, the application satisfies usability criteria. The participants were, in general, satisfied with the functionalities and visual appearance of the application. Most of the participants were successfully able to complete most of the tasks involved in the evaluation.

There is still much space for improvement in usability and shortening the learning curve for new users as per the feedback and suggestion from participants. The participants also suggested providing a UI guide and a help feature for new users as the application can be overwhelming. The participants and users in general also need more time to get familiar with the system. As most participants are not native English speakers, this could have affected their experience and response. Some of

the participants also suggested that the evaluation was very long and must be reduced. As most of the participants were professionals and had a limited time for the evaluation, these circumstances could have influenced their overall experience. Also, only 68 % (15) of the evaluation participants were familiar with an e-learning tool or learning software. Participant's prior experiences could have also affected their response during the evaluation. Additionally, the participants are not too familiar with the system and therefore are not aware of how informative specific learning metrics could be.

The evaluation had 22 participants only, among whom there were only four professional educators, which could have also affected the evaluation results. Additionally, the evaluation included NASA-RTLX on 5 points likert scale as a short version of NASA-TLX, which could have affected the perceived workload evaluation.

Additionally, participant's familiarity with the sCool learning platform could also have affected their evaluation and experience. The participant's usage of the application over an entire course or a period would have been beneficial.

Factors such as evaluation topic, length of the evaluation, ordering of questions, and using the English language for the survey could also have had an immense impact on the participants (Fan and Yan, 2010). Additionally, the COVID-19 pandemic has affected everyone around the world in multiple ways, ranging from personal losses and other mental health issues (Kontoangelos et al., 2020; Gavin et al., 2020). These issues could have also affected the participant's overall experience with the application and evaluation.

Chapter 7

Lesson Learned

In this chapter, the experiences and issues encountered during the various project phases are described. The conclusions about the literature, development, and outcomes are also included.

7.1 Literature

One of the significant parts of further development of the sCool environment was introducing additional learning analytics for the sCool system. Various game-based applications are currently available for learning programming skills or computation skills for students or novice learners of various age groups. However, regardless of age, a novice programming learner could face many difficulties while attempting to understand and learn coding concepts (Lahtinen et al., 2005). These difficulties have to be acknowledged and recognized in order to be able to assist the students (Albluwi and Salter, 2020). Hence, these applications also provide learning analytics capabilities to their various stakeholders, comprising educators, parents, students, and game developers. Related papers and studies were considered for a comprehensive overview of these tools and their capabilities to end-users (Freire et al., 2016; Owen and Baker, 2019). The relevant data searched was gathered, and the tools were analyzed to perform an evaluation and identify their relevance for this work

(CodeMonkey, 2021; Ozaria, 2021; Steinmaurer et al., 2020).

The features of these learning analytics tools may vary from one tool to another. Different stakeholders may have different requirements and interests, such as students being more interested in identifying their course progress. In contrast, educators may want to analyze the learning progress of their entire class, and game developers may want feedback to improve the game content or game design or finding strategies used by students for course completion. Hence the tools also provide varying features for the different stakeholders. Some of the tools focus on teachers and provide a minimalist course progress dashboard for the students (CodeMonkey, 2021; Ozaria, 2021). These tools were analyzed and considered in the concept for developing a learning analytics application for sCool.

For analysis of student’s game-related data concerning learning coding or computation skills, literature in code analysis was considered (Albluwi and Salter, 2020). It is also essential to find the effectiveness of course content in making students understand and learn the intended programming concepts. Analyzing the submitted program solutions provides an overview of the programming concepts used by students in completing a programming task and provides insights into whether students were able to learn and use various programming concepts.

Various literature was also considered to understand data visualization techniques to convey information in simple and understandable ways to end-users. Some case studies also provide information about various data visualization tools and frameworks and their advantages and disadvantages, respectively (J. Liu et al., 2018). In the end, the tool and framework which would fit the system and available resources were considered. Lower dimensional data visualization techniques such as one-dimensional, two-dimensional, or three-dimensional plots were considered to convey information understandably and simplistically.

7.2 Development

The sCool platform consisted of two components, a mobile video game and a web application, which communicated with each other using REST APIs. The game has

two game modes as the concept-learning theory mode and the programming-learning practical mode. In the concept-learning theory mode, students learn programming concepts, and they utilize these concepts to solve programming tasks in the practice mode. The player's game-related data is collected and stored in the sCool's MySQL database instance. The students can also see their course progress in the mobile game itself.

The main focus of this thesis was to provide features of learning analytics to stakeholders of educators and game developers of the sCool system. A sCool learning analytics application was proposed and developed to achieve this goal. The application was developed using Plotly's Dash framework, which is an open-source full-stack framework. Plotly's Dash framework also provides various data visualization capabilities. Data analysis of student's data is performed using Python. The results are presented to the users using intuitive and straightforward UI. Docker¹ image was used for a fast, easy, and portable application development and to deploy the application on a server.

7.3 Outcome

The evaluation process considered various aspects of the application. For the evaluation, it was necessary to make the application reliable and publicly available for participants to participate in the evaluation and complete it. A docker image of the application was deployed on Google Cloud Platform (GCP)² which currently provides a three-month free usage plan for developers (with credits on GCP). The application connects to the sCool database to access player's game-related data. Data of only those players are used in the application, who had provided explicit permission in the sCool mobile game to permit processing of their game-related data. The application was introduced using an audio-video recording, and the recording was made available in the initial introductory steps of the evaluation.

The results of the evaluation showed that the participants well received the ap-

¹<https://www.docker.com/>

²<https://cloud.google.com/>

plication. The participants were able to identify a player who is performing poorly on game metrics such as the number of tasks completed, the number of lines of code submitted, progress in the course, points collected, and tasks attempted. The participants provided a positive usability rating which implies that the application satisfies usability criteria. However, based on the feedback and suggestion provided, further investigation and improvements in the user interface are required to improve the application's usability and shorten the learning curve experienced by new users. A help UI guide could also be provided to lower the learning curve for new users.

Chapter 8

Conclusion

In this chapter, the conclusion of the key elements and characteristics of the project are described. An outlook for future improvements in learning analytics capabilities for the coding learning game sCool is also discussed.

8.1 Conclusion

The current Coronavirus disease 2019 (COVID-19) pandemic has forced widespread school closures, forcing students to learn from home. The pandemic forcing students to learn from home has emphasized the importance of effective remote learning. Learning games provide an engaging means of effective learning. Many coding learning applications are available to all age groups in various modes such as games and web applications. As learning programming or computational skill can be a challenging task for novice learners of any age group, teachers and game developers need to identify the progress made by students and analyze player's game-related data for providing timely assistance to students and making improvements in course content and game design. The project aimed to introduce a learning analytics application for the sCool coding learning platform.

In the first phase of the project, various coding learning applications, including sCool, and their learning analytics applications, were analyzed. The requirements

were considered, and concepts were developed. Key elements of the analysis were developing a learning analytics application for sCool stakeholders comprising of educators and game developers. The players on sCool could already view their course progress in-game.

In the second phase of the project, the sCool learning analytics application was evaluated by 22 volunteers, and their feedbacks were collected. The participants were able to identify poor-performing students based on various game metrics such as course progress, programming solutions submitted, coding concepts learned, and tasks completed. Most of the participants were able to identify tasks that were the most completed or least completed tasks and responded positively concerning identifying challenging or easy tasks. The participants also responded positively by providing positive usability ratings on SUS and perceived workload ratings on NASA-TLX.

The sCool learning analytics application is a web-based full-stack application developed using open-source frameworks. The learning analytics web application is intended to be used by educators and sCool game developers to analyze students' performance on the platform. The educators and game developers could utilize their newfound insights from the learning analytics application to assist students in need of support, improve course content, and improve game design.

8.2 Future Work

There is a need for additional improvements in the application's usability and to shorten the learning curve for new users. The future work could provide an onboarding tutorial for new users. The evaluation results also showed some critical areas of navigation, user interface, and a help UI guide. Most of the feedback suggested that the application can be overwhelming for new users and has a learning curve of its own. The navigation could be improved, and fewer and more relevant data could be shown to educators to overview students of their class quickly.

The participants also suggested an improvement in filtering in the overview table. The custom plots or plots, in general, have a short screen space and hence can

have overlapping information or miss out on some essential information entirely. The missing information results in confusion and frustration. Although this is a device screen issue, possible improvements must be further analyzed. The tasks with a high and low success and fail number can be highlighted to provide quick information. The expand buttons could be a bit separated from the background using CSS improvements.

The application should also be able to support further modification and improvements in the sCool game environment. New features added in the database should also either automatically be supported in the application or require minimal effort.

The future work could also provide learning analytics capabilities to players/students about their performance. The players/students could also identify their progress compared to their peers or their own learning goals.

Bibliography

- Abt, C. C. (1981). *Serious games*. The Viking Press.
- Albluwi, I., & Salter, J. (2020). Using static analysis tools for analyzing student behavior in an introductory programming course. *Jordanian Journal of Computers and Information Technology (JJCIT)*, *6*, 215–233. <https://doi.org/10.5455/jjcit.71-1584234700>
- Ali, S. M., Gupta, N., Nayak, G. K., & Lenka, R. K. (2016). Big data visualization: Tools and challenges. *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, 656–660. <https://doi.org/10.1109/IC3I.2016.7918044>
- Alonso-Fernandez, C., Calvo, A., Freire, M., Martinez-Ortiz, I., & Fernandez-Manjon, B. (2017). Systematizing game learning analytics for serious games. *2017 IEEE Global Engineering Education Conference (EDUCON)*, 1111–1118. <https://doi.org/10.1109/EDUCON.2017.7942988>
- Arnab, S., Lim, T., Carvalho, M., Bellotti, F., Freitas, S., Louchart, S., Suttie, N., Berta, R., & De Gloria, A. (2014). Mapping learning and game mechanics for serious games analysis: Mapping learning and game mechanics. *British Journal of Educational Technology*. <https://doi.org/10.1111/bjet.12113>
- Bachvarova, Y., Bocconi, S., van der Pols, B., Popescu, M., & Roceanu, I. (2012). Measuring the effectiveness of learning with serious games in corporate training. *Procedia computer science*, *15*, 221–232.
- Bangor, Aaron, Kortum, P., T., P., Miller, & T., J. (2008). The system usability scale (sus): An empirical evaluation. *International Journal of Human-Computer Interaction*, *24*, 574–. <https://doi.org/10.1080/10447310802205776>

- Bangor, A., Kortum, P., & Miller, J. (2009). Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3), 114–123.
- Baojiang Cui, Jiansong Li, Tao Guo, Jianxin Wang, & Ding Ma. (2010). Code comparison system based on abstract syntax tree. *2010 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT)*, 668–673. <https://doi.org/10.1109/ICBNMT.2010.5705174>
- Bar chart. (2021). Bar chart — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Bar_chart (accessed 10.04.2021)
- Barr, V., & Stephenson, C. (2011). Bringing computational thinking to k-12: What is involved and what is the role of the computer science education community? *ACM Inroads*, 2. <https://doi.org/10.1145/1929887.1929905>
- Bates, A. W. (1997). The impact of technological change on open and distance learning. *Distance education*, 18(1), 93–109.
- Bellotti, F., Kapralos, B., Lee, K., & Moreno Ger, P. (2013). User assessment in serious games and technology-enhanced learning. *Advances in Human-Computer Interaction*, 2013. <https://doi.org/10.1155/2013/120791>
- Bidarra, J. (2009). Emerging digital media, games and simulations: A challenge for open and distance learning. *Revista de Ciências da Computação*, 4(4), 1–15.
- Borah, R. R. (2013). Slow learners: Role of teachers and guardians in honing their hidden skills. *International Journal of Educational Planning & Administration*, 3(2), 139–143.
- Börner, K., Bueckle, A., & Ginda, M. (2019). Data visualization literacy: Definitions, conceptual frameworks, exercises, and assessments. *Proceedings of the National Academy of Sciences*, 116(6), 1857–1864.
- Bostock, M. (2014). D3.js - let's make a bubble map. <https://bost.ocks.org/mike/bubble-map/>
- Bostock, M. (2021). Chart.js - sample bubble chart. <https://www.chartjs.org/docs/latest/samples/other-charts/bubble.html>
- Brooke, J. (1995). Sus: A quick and dirty usability scale. *Usability Eval. Ind.*, 189.

BIBLIOGRAPHY

- CodeMonkey. (2021). Codemonkey.com. *CodeMonkey Studios*. <https://app.codemonkey.com/> (accessed: 23.02.2021)
- Colligan, L., Potts, H. W., Finn, C. T., & Sinkin, R. A. (2015). Cognitive workload changes for nurses transitioning from a legacy system with paper documentation to a commercial electronic health record. *International Journal of Medical Informatics*, *84*(7), 469–476. <https://doi.org/https://doi.org/10.1016/j.ijmedinf.2015.03.003>
- Combéfis, S., Beresnevičius, G., & Dagiene, V. (2016). Learning programming through games and contests: Overview, characterisation and discussion. *10*, 39–60. <https://doi.org/10.15388/ioi.2016.03>
- Comparision of JavaScript Charting Libraries. (2021). Comparision of javascript charting libraries — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Comparison_of_JavaScript_charting_libraries (accessed 21.02.2021)
- Computer Science Teachers Association. (2021). Computer science teachers association. <https://www.csteachers.org/> (accessed: 24.02.2021)
- Cox, B. J. (1986). Object-oriented programming: An evolutionary approach.
- Dash. (2021). Dash introduction. *Plotly*. <https://dash.plotly.com/introduction> (accessed: 16.04.2021)
- Department of Education, U. S. (2010a). National education technology plan. <http://www.ed.gov/technology/netp-2010>
- Department of Education, U. S. (2017). Reimagining the role of technology in education: National education technology plan. <https://tech.ed.gov/netp/>. (accessed: 21.01.2021)
- Dietz-Uhler, B., & Hurn, J. E. (2013). Using learning analytics to predict (and improve) student success: A faculty perspective. *Journal of interactive online learning*, *12*(1), 17–26.
- Dillenbourg, P., Zufferey, G., Alavi, H., Jermann, P., Do-Lenh, S., & Bonnard, Q. (2011). Classroom orchestration: The third circle of usability.
- Djaouti, D., Alvarez, J., Jessel, J. P., & Rampnoux, O. (2011). Origins of serious games. *Serious Games and Edutainment Applications*, 25–43.

BIBLIOGRAPHY

- Drachen, A., El-Nasr, M. S., & Canossa, A. (2013a). Game analytics – the basics. *Game Analytics: Maximizing the Value of Player Data*, 13–40. https://doi.org/10.1007/978-1-4471-4769-5_2
- Drachen, A., El-Nasr, M. S., & Canossa, A. (2013b). Game analytics—the basics. *Game analytics* (pp. 13–40). Springer.
- Fan, W., & Yan, Z. (2010). Factors affecting response rates of the web survey: A systematic review. *Computers in human behavior*, 26(2), 132–139.
- Fan Bao, & Jia Chen. (2014). Visual framework for big data in d3.js. *2014 IEEE Workshop on Electronics, Computer and Applications*, 47–50. <https://doi.org/10.1109/IWECA.2014.6845553>
- Few, S. (2006). *Information dashboard design: The effective visual communication of data*. O'Reilly.
- Fields, T. V. (2013). Game industry metrics terminology and analytics case study. In M. Seif El-Nasr, A. Drachen, & A. Canossa (Eds.), *Game analytics: Maximizing the value of player data* (pp. 53–71). Springer London. https://doi.org/10.1007/978-1-4471-4769-5_4
- Flask login. (2021). Flask-login. *Flask*. <https://flask-login.readthedocs.io/en/latest/> (accessed: 16.04.2021)
- Freire, M., Serrano-Laguna, Á., Manero, B., Martinez-Ortiz, I., Moreno Ger, P., & Fernández-Manjón, B. (2016). Game learning analytics: Learning analytics for serious games. https://doi.org/10.1007/978-3-319-17727-4_21-1
- Gantt, H. L. (1910). Work, wages and profit. *The Engineering Magazine*, 215–233. re-published as *Work, Wages and Profits*, Easton, Pennsylvania, Hive Publishing Company, 1974.
- Gasevic, D., Pardo, A., Dawson, S., & Steigler-Peters, S. (2016). The role of learning analytics in future education models. <https://doi.org/10.13140/RG.2.2.16834.68802>
- Gavin, B., Lyne, J., & McNicholas, F. (2020). Mental health and the covid-19 pandemic. *Irish journal of psychological medicine*, 37(3), 156–158.
- graph, R. (2021). Raw graph - how to make a scatterplot. <https://rawgraphs.io/learning/how-to-make-a-scatterplot/>

BIBLIOGRAPHY

- Grover, S., & Pea, R. (2013). Computational thinking in k–12 a review of the state of the field. *Educational Researcher*, *42*, 38–43. <https://doi.org/10.3102/0013189X12463051>
- Hart, S. G. (2006). Nasa-task load index (nasa-tlx); 20 years later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *50*(9), 904–908. <https://doi.org/10.1177/154193120605000909>
- Hossain, S., Calloway, C., Lippa, D., Niederhut, D., & Shupe, D. (2019). Visualization of bioinformatics data with dash bio. *Proceedings of the 18th Python in Science Conference*, 126–133.
- Hui, S. K. (2013). Gamer retention in social games using aggregate dau and mau data : A bayesian data augmentation approach.
- Infogram. (2021a). Infogram. <https://infogram.com/>
- Infogram. (2021b). Infogram sample dashboard. <https://infogram.com/teal-light-1hmr6g78x5djz6n>
- International Society for Technology in Education. (2021). International society for technology in education. *ISTE*. <https://www.iste.org/> (accessed: 24.02.2021)
- Jarrell, S. (1994). *Basic statistics*. Wm. C. Brown Pub. <https://books.google.at/books?id=Jn5GAAAAYAAJ>
- Junaidi, J., Julianto, A., Anwar, N., Safrizal, Spits Warnars, H. L. H., & Hashimoto, K. (2018). Perfecting a video game with game metrics. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, *16*, 1324–1331. <https://doi.org/10.12928/TELKOMNIKA.v16i3.7209>
- Kamiya, T., Kusumoto, S., & Inoue, K. (2002). Ccfinder: A multilinguistic token-based code clone detection system for large scale source code. *Software Engineering, IEEE Transactions on*, *28*, 654–670. <https://doi.org/10.1109/TSE.2002.1019480>
- Kanaki, K., & Kalogiannakis, M. (2018). Introducing fundamental object-oriented programming concepts in preschool education within the context of physical science courses. *Education and Information Technologies*, *23*. <https://doi.org/10.1007/s10639-018-9736-0>

- Keim, D. A. (2002). Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1), 1–8. <https://doi.org/10.1109/2945.981847>
- Khalil, M., & Ebner, M. (2015). Learning analytics: Principles and constraints. <https://doi.org/10.13140/RG.2.1.1733.2083>
- Kiili, K., Moeller, K., & Ninaus, M. (2018). Evaluating the effectiveness of a game-based rational number training - in-game metrics as learning indicators. *Computers I& Education*, 120, 13–28. <https://doi.org/https://doi.org/10.1016/j.compedu.2018.01.012>
- Klein, R. (2001). *Scheduling of resource constrained projects*. Kluwer Academic Publishers.
- Kojic, A. (2017). *Design and implementation of an adaptive multidisciplinary educational mobile game* (Master's thesis). Graz University of Technology. Graz, Austria.
- Kojic, A., Koji, M., Pirker, J., Gütl, C., Mentzelopoulos, M., & Economo, D. (2018). Scool - a mobile flexible learning environment. <https://doi.org/http://dx.doi.org/10.1002/andp.19053221004>. (accessed: 24.01.2021)
- Kojic, M. (2017). *Procedural content generation in a multidisciplinary educational mobile game* (Master's thesis). Graz University of Technology. Graz, Austria.
- Kontoangelos, K., Economou, M., & Papageorgiou, C. (2020). Mental health effects of covid-19 pandemia: A review of clinical and psychological traits. *Psychiatry investigation*, 17(6), 491.
- Kosara, R. (2016). Presentation-oriented visualization techniques. *IEEE Computer Graphics and Applications*, 36(1), 80–85. <https://doi.org/10.1109/MCG.2016.2>
- Koster, R. (2004). *Theory of fun for game design*. Paraglyph Press.
- Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, 37, 14–18. <https://doi.org/10.1145/1067445.1067453>

BIBLIOGRAPHY

- Leff, A., & Rayfield, J. T. (2001). Web-application development using the model/view/controller design pattern. *Proceedings fifth ieee international enterprise distributed object computing conference*, 118–127.
- Lewis, J., & Sauro, J. (2018). Item benchmarks for the system usability scale. *13*, 158–167.
- Li, I., Dey, A., & Forlizzi, J. (2010). A stage-based model of personal informatics systems. *Conference on Human Factors in Computing Systems - Proceedings*, *1*, 557–566. <https://doi.org/10.1145/1753326.1753409>
- Likert, R. (1932). *A technique for the measurement of attitudes*. Archives of Psychology. <https://books.google.at/books?id=9rotAAAAYAAJ>
- Lim, T., Louchart, S., Suttie, N., Ritchie, J., Aylett, R., Stefan, I., Ion, R., Martinez-Ortiz, I., & Moreno Ger, P. (2013). Strategies for effective digital games development and implementation. <https://doi.org/10.4018/978-1-4666-2848-9.ch010>
- Liu, J., Tang, T., Wang, W., Xu, B., Kong, X., & Xia, F. (2018). A survey of scholarly data visualization. *IEEE Access*, *6*, 19205–19221. <https://doi.org/10.1109/ACCESS.2018.2815030>
- Liu, S., Gao, C., Chen, S., Lun Yiu, N., & Liu, Y. (2020). Atom: Commit message generation based on abstract syntax tree and hybrid ranking. *IEEE Transactions on Software Engineering*, 1–1. <https://doi.org/10.1109/TSE.2020.3038681>
- Long, P., & Siemens, G. (2011). Penetrating the fog : Analytics in learning and education. <https://er.educause.edu/~media/files/article-downloads/erm1151.pdf>. (accessed: 24.01.2021)
- Mease, J. (2018). Bringing ipywidgets support to plotly. py. *Proceedings of the 17th Python in Science Conference (SciPy 2018), Austin, TX, USA*, 9–15.
- Mellon, L. (2009). *Applying metrics driven development to mmo costs and risks* (tech. rep.).
- Michael, D. R., & Chen, S. L. (2006). Serious games: Games that educate, train, and inform.
- Miller, S. (2001). Workload measures. *National Advanced Driving Simulator. Iowa City, United States*.

BIBLIOGRAPHY

- Mitchell, J. C., & Apt, K. (2003). *Concepts in programming languages*. Cambridge University Press.
- Moroney, W., Biers, D., Eggemeier, F., & Mitchell, J. (1992). A comparison of two scoring procedures with the nasa task load index in a simulated flight task. *Proceedings of the IEEE 1992 National Aerospace and Electronics Conference (at NAECON) 1992*, 734–740 vol.2. <https://doi.org/10.1109/NAECON.1992.220513>
- Mosquera, C., Steinmaurer, A., Eckhardt, C., & Guetl, C. (2020). Immersively learning object oriented programming concepts with scool, 124–131. <https://doi.org/10.23919/iLRN47897.2020.9155144>
- mouseover event. (2021). Mouseover event. https://developer.mozilla.org/en-US/docs/Web/API/Element/mouseover_event (accessed 30.04.2021)
- Muppudathi, G. (2014). Role of teachers on helping slow learners to bring out their hidden skills. *International journal of scientific research*, 3(3).
- NaN. (2021). Nan — missing data. https://pandas.pydata.org/pandas-docs/stable/user_guide/missing_data.html (accessed: 16.04.2021)
- NASA, N. (1986). Task load index (tlx) v. 1.0 manual. *NASA, NASA-Ames Research Center Moffett Field*.
- ODBC overview. (2021). Odbc overview. <https://docs.microsoft.com/en-us/sql/odbc/reference/odbc-overview> (accessed: 16.04.2021)
- Owen, V., & Baker, R. (2019). Learning analytics for games.
- Ozaria. (2021). Codecombat ozaria. *CodeCombat Inc*. <https://www.ozaria.com/> (accessed: 24.02.2021)
- Pandas. (2021). Pandas. *Pandas*. <https://pandas.pydata.org/docs/index.html> (accessed: 16.04.2021)
- Pandas DataFrame. (2021). Pandas dataframe. *Pandas*. <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html> (accessed: 16.04.2021)
- Pandas groupby. (2021). Pandas dataframe groupby. <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.groupby.html> (accessed: 16.04.2021)
- Panichella, A., Dit, B., Oliveto, R., Di Penta, M., Poshynanyk, D., & Lucia, A. (2013). How to effectively use topic models for software engineering tasks? an

- approach based on genetic algorithms. *Proceedings - International Conference on Software Engineering*. <https://doi.org/10.1109/ICSE.2013.6606598>
- Patino, A., Romero, M., & Proulx, J.-N. (2016). Analysis of game and learning mechanics according to the learning theories. <https://doi.org/10.1109/VS-GAMES.2016.7590337>
- Plass, J. L., Homer, B. D., Kinzer, C. K., Chang, Y. K., Frye, J., Kaczetow, W., Isbister, K., & Perlin, K. (2013). Metrics in simulations and games for learning. *Game analytics* (pp. 697–729). Springer.
- Plotly. (2021). Plotly python mixed subplot. <https://plotly.com/python/mixed-subplots/>
- Python ast. (2021). Python standard library - ast module. *Python*. <https://docs.python.org/3/library/ast.html> (accessed: 16.04.2021)
- Python json. (2021). Python standard library - json module. *Python*. <https://docs.python.org/3/library/json.html> (accessed: 16.04.2021)
- RAWGraphs. (2021). Raw graphs. <https://rawgraphs.io/>
- Richardson, J., Tapadinhas, J., Carlie, J., & Idoine. (2018). Magic quadrant for analytics and business intelligence platforms.
- Rogozhkina, I., & Kushnirenko, A. (2011). Piktomir: Teaching programming concepts to preschoolers with anew tutorial environment. *Procedia - Social and Behavioral Sciences*, 28, 601–605. <https://doi.org/10.1016/j.sbspro.2011.11.114>
- Sadiku, M., Shadare, A., Musa, S., Akujuobi, C., & Perry, R. (2016). Data visualization. *International Journal of Engineering Research and Advanced Technology (IJERAT)*, 12, 2454–6135.
- Sajnani, H., Saini, V., Svajlenko, J., Roy, C., & Lopes, C. (2015). Sourcerercc: Scaling code clone detection to big code.
- Scatter plot. (2021). Scatter plot — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Scatter_plot (accessed 10.04.2021)
- Schmillen, C. (2014). Rock paintings from the cave of beasts (gilf kebir, libyan desert) estimated 7000 bp [File: **Bestias11.JPG**]. <https://commons.media.org/w/index.php?curid=31399425>

- Seif El-Nasr, M., & Canossa, A. (2013). Interview with jim baer and daniel mccaffrey from zynga. In M. Seif El-Nasr, A. Drachen, & A. Canossa (Eds.), *Game analytics: Maximizing the value of player data* (pp. 73–82). Springer London. https://doi.org/10.1007/978-1-4471-4769-5_5
- Siirtola, H. (2014). "bars, pies, doughnuts & tables – visualization of proportions", 240–245. <https://doi.org/10.14236/ewic/hci2014.30>
- Sommerville, I. (2010). *Software engineering* (9th ed.). Addison-Wesley.
- The state of online gaming – 2018* (tech. rep.). (2018). Limelight Networks. <https://www.limelight.com/blog/state-of-online-gaming-2018/> (accessed: 21.01.2021)
- The state of online gaming – 2020* (tech. rep.). (2020). Limelight Networks. <https://www.limelight.com/resources/white-paper/state-of-online-gaming-2020/> (accessed: 21.01.2021)
- State of online gaming 2021* (tech. rep.). (2021). Limelight Networks. <https://www.limelight.com/blog/state-of-online-gaming-2021/> (accessed: 21.01.2021)
- Stefik, M., & Bobrow, D. G. (1985). Object-oriented programming: Themes and variations. *AI magazine*, 6(4), 40–40.
- Steinmaurer, A. (2019). *Revising a game-based learning platform for computational skills in education* (Master's thesis). Graz University of Technology. Graz, Austria.
- Steinmaurer, A., Pirker, J., & Guetl, C. (2020). Scool - game based learning in stem education: A case study in secondary education. https://doi.org/10.1007/978-3-030-11932-4_58
- Steinmaurer, A., Pirker, J., & Gütl, C. (2019). Scool - game-based learning in computer science class a case study in secondary education. *International Journal of Engineering Pedagogy*, 9(2), 26–50. <https://doi.org/10.3991/ijep.v9i2.9942>
- Strachey, C. (2000). Fundamental concepts in programming languages. *Higher-order and symbolic computation*, 13(1), 11–49.
- tableau. (2021a). Global covid-19 tracker. https://public.tableau.com/profile/covid.19.data.resource.hub#!/vizhome/COVID-19Cases_15840488375320/COVID-19GlobalView

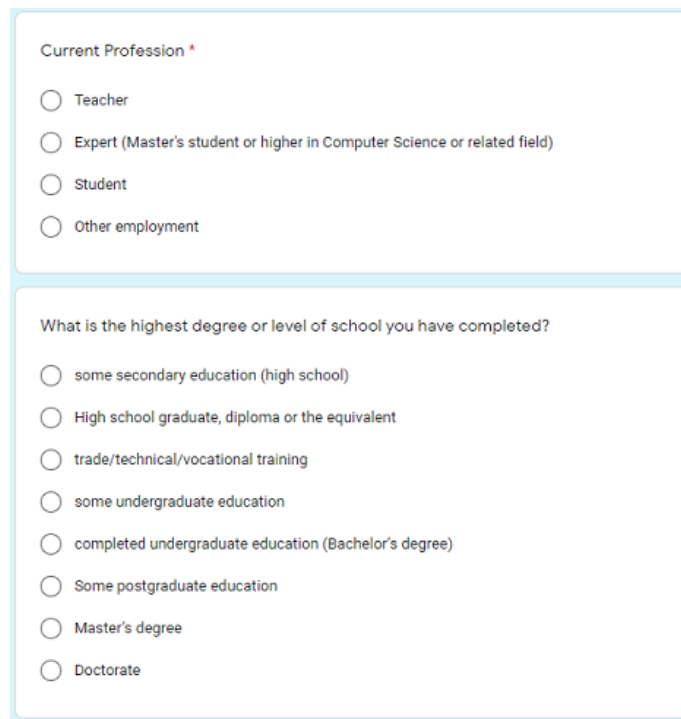
BIBLIOGRAPHY

- tableau. (2021b). What is tableau. <https://www.tableau.com/why-tableau/what-is-tableau>
- Telea, A. C. (2014). *Data visualization: Principles and practice*. CRC Press.
- UNESCO. (2021). Covid-19 impact on education. *UNESCO*. <https://en.unesco.org/covid19/educationresponse> (accessed: 18.01.2021)
- Verbert, K., Duval, E., Klerkx, J., Govaerts, S., & Santos, J. L. (2013). Learning analytics dashboard applications. *American Behavioral Scientist*, 57(10), 1500–1509. <https://doi.org/10.1177/0002764213479363>
- Ward, M. O., Grinstein, G., & Keim, D. (2010). *Interactive data visualization: Foundations, techniques, and applications*. CRC Press.
- Wing, J. (2006). Computational thinking. *Communications of the ACM*, 49, 33–35. <https://doi.org/10.1145/1118178.1118215>
- Zhang, J., Wang, X., Zhang, H., Sun, H., Wang, K., & Liu, X. (2019). A novel neural source code representation based on abstract syntax tree. *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, 783–794. <https://doi.org/10.1109/ICSE.2019.00086>
- Zhou, J., Zhang, H., & Lo, D. (2012). Where should the bugs be fixed? more accurate information retrieval-based bug localization based on bug reports. *Proceedings - International Conference on Software Engineering*, 14–24. <https://doi.org/10.1109/ICSE.2012.6227210>

Appendix A

Questionnaire

A1 Intro



Current Profession *

- Teacher
- Expert (Master's student or higher in Computer Science or related field)
- Student
- Other employment

What is the highest degree or level of school you have completed?

- some secondary education (high school)
- High school graduate, diploma or the equivalent
- trade/technical/vocational training
- some undergraduate education
- completed undergraduate education (Bachelor's degree)
- Some postgraduate education
- Master's degree
- Doctorate

Figure A.1: Participants intro

APPENDIX A. QUESTIONNAIRE

Are you familiar with learning software Or e-learning tools ?

✓ Yes

✗ No

? Somewhat

Figure A.2: Participants familiarity with e-learning tools

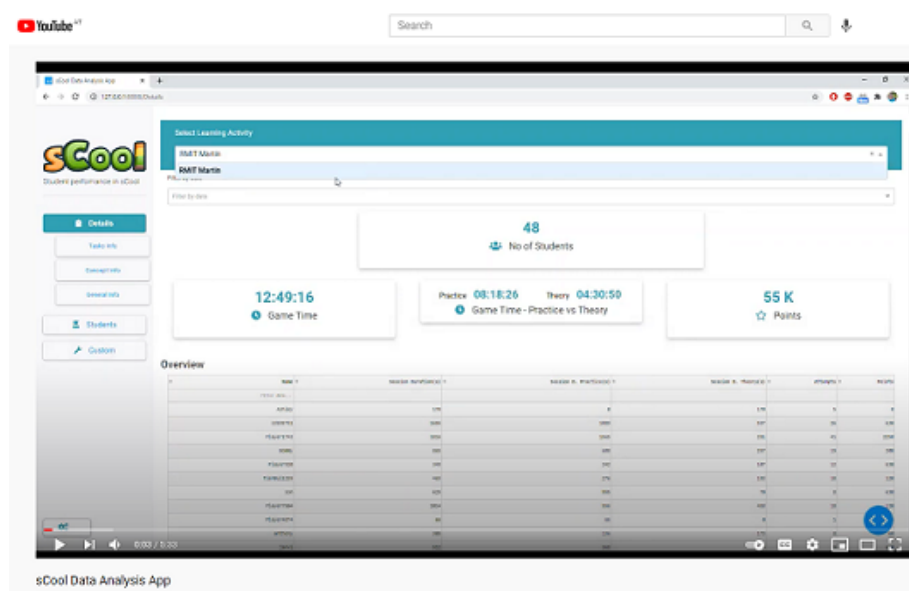


Figure A.3: Intro pre-recorded audio-video

A2 Tasks

Which Practice task was completed by least number of students in the entire class? 1 point

Hint: Details -> Task information - No. of students completing a Task - table or graphs

- Calculate the Fibonacci Sequence and check it with the storage
- The first task will be to move Rob!
- Collect the coins and print Rob's storage

Figure A.4: Task 1

Which Theory task was completed by least number of students in the entire class? 1 point

Hint: Details -> Task information - No. of students completing a Task - table or graphs

- Which of the following is a python program broken down into?
- What is a list in Python?
- What is the definition of a function?

Figure A.5: Task 2

How many Theory tasks did student Claire solve 1 point

Hint: Students -> No. of tasks completed

- 6
- 8
- 5

Figure A.6: Task 3

APPENDIX A. QUESTIONNAIRE

Which of the below students spent least game time (session duration) in the entire class 1 point

Hint: Details -> Overview

- pattt
- Thomas
- claire
- Alexander

Figure A.7: Task 4

Which student completed least number of Theory tasks in the entire class 1 point

Hint: Details -> Task info -> (Theory) Count of tasks completed by students

- claire
- s3779710 VincentC
- Thomas
- Alexander

Figure A.8: Task 5

APPENDIX A. QUESTIONNAIRE

claire used which concepts (select all which were used)

1 point

Hint: Students tab

- Variable
- Constant
- Function call
- Compare
- Assignment

Figure A.9: Task 6

Thomas used which concepts (select all which were used)

1 point

Hint: Students tab

- Variable
- For loop
- Function call
- Compare
- Assignment

Figure A.10: Task 7

Which skills did student Alexander complete

1 point

Hint: Students tab

- Introduction
- Data Types
- Constructs and Logic

Figure A.11: Task 8

APPENDIX A. QUESTIONNAIRE

Did Thomas complete the course 'RMIT_2020' ?

1 point

Hint: Students tab

- Yes
- No
- Maybe

Figure A.12: Task 9

How many lines of code did Ashley write during the course?

1 point

Hint: Students tab - Count of lines of Code

- 10
- 0
- 27

Figure A.13: Task 10

To find which student has the most Syntax errors - make a custom plot of 'Syntax Errors in runs (No. of times)' vs 'Student'. Which student has most syntax error in the entire class?

1 point

Hint: Custom -> group by Student; feature 1: Syntax Errors in runs (No. of times); feature 2 : Student; Bar or Scatter plot

- son (son-677)
- vidodi99 (vidodi99-733)
- Thomas (Thomas-699)

Figure A.14: Task 11

A3 Application Specific Questions

Would you use the tool for analysis of your students?

✓ Yes

✗ No

? Maybe

Were you able to identify the progress of a student in the course using the tool?
For example Thomas completed 28% of the course RMIT_2020 and Alexander completed ~70% of the course RMIT_2020

✓ Yes

✗ No

? Maybe

Were you able to identify students who are performing poorly and may need assistance using the tool? *
For example No code written, less progress in the course, many wrong attempts, and few tasks completed.

✓ Yes

✗ No

? Maybe

Were you able to identify tasks that were easy or hard for students? (For example, the most completed tasks or tasks which no one completed.) *
This knowledge could be used to improve course content by improving them or creating new relevant tasks.

✓ Yes

✗ No

? Maybe

Figure A.15: Participants educators I

APPENDIX A. QUESTIONNAIRE

Is this tool meaningful?

✓ Yes

✗ No

? Maybe

Rate the usability

1 2 3 4 5

😡 Not useable 😊 Very useable

Is all the data collected and analyzed relevant for Education purpose or useful for analyzing students progress?

✓ Yes

✗ No

? Maybe

Would you recommend this to a colleague?

😊 Yes

😞 No

😐 Maybe

Feedback
Example: Usability or data collected and analyzed or any other suggestions for improvement as a user

Your answer _____

Figure A.16: Participants educators II

APPENDIX A. QUESTIONNAIRE

Were you able to identify the progress of a student in the course using the tool?
For example Thomas completed 28% of the course RMIT_2020 and Alexander completed ~70% of the course RMIT_2020

✓ Yes

✗ No

? Maybe

Were you able to identify students who are performing poorly and may need assistance using the tool? *

For example No code written, less progress in the course, many wrong attempts, and few tasks completed.

✓ Yes

✗ No

? Maybe

Were you able to identify tasks that were easy or hard for students? (For example, the most completed tasks or tasks which no one completed.) *

This knowledge could be used to improve course content by improving them or creating new relevant tasks.

✓ Yes

✗ No

? Maybe

Would you recommend this to a colleague?

😊 Yes

😐 No

😞 Maybe

Feedback
Example: Usability or data collected and analyzed or any other suggestions for improvement as a user

Your answer _____

Figure A.17: Participants except educators

A4 NASA-TLX

Mental Demand
How mentally demanding was the task? Was the task easy or demanding, simple or complex?

1 2 3 4 5

😊 Very low 😞 Very high

Temporal Demand
How much time pressure did you feel due to the pace at which the tasks or task elements occurred? Was the pace slow or rapid?

1 2 3 4 5

😊 Very low 😞 Very high

Overall Performance
How successful were you in performing the task? How satisfied were you with your performance?

1 2 3 4 5

😊 Very low 😞 Very high

Effort
How hard did you have to work to accomplish your level of performance?

1 2 3 4 5

😊 Very low 😞 Very high

Frustration Level
How irritated, stressed, and annoyed versus content, relaxed, and complacent did you feel during the task?

1 2 3 4 5

😊 Very low 😞 Very high

Figure A.18: NASA-TLX

A5 System Usability Scale

Table A.1: System Usability Scale

| | Strongly Disagree | Disagree | Undecided | Agree | Strongly Agree |
|--|------------------------------|-----------------|------------------|--------------|---------------------------|
| I think I would like to use this application frequently | 0 | 0 | 0 | 0 | 0 |
| I think I found the application unnecessary complex | 0 | 0 | 0 | 0 | 0 |
| I thought this application was easy to use | 0 | 0 | 0 | 0 | 0 |
| I think that I would need the help of a technical person to use this application | 0 | 0 | 0 | 0 | 0 |
| I found the various functions in this application were well integrated | 0 | 0 | 0 | 0 | 0 |
| I thought there was too much inconsistency in the application | 0 | 0 | 0 | 0 | 0 |
| I would imagine that most people would learn to use this application very quickly | 0 | 0 | 0 | 0 | 0 |
| I found this application very awkward to use | 0 | 0 | 0 | 0 | 0 |
| I felt very confident using this application | 0 | 0 | 0 | 0 | 0 |
| I needed to learn a lot of things before I could get going with this application | 0 | 0 | 0 | 0 | 0 |

