Michael Spitzer

Teamsketch

Fo(u)r drawers - fo(u)r iPad(s) - one collaborative sketch

Master's Thesis

Graz University of Technology

Institute of Information Systems and Computer Media (IICM)
Head: Prof. Dipl.-Ing. Dr.techn. Frank Kappe

Supervisor: Assoc. Prof. Dipl.-Ing. Dr.techn. Martin Ebner

Graz, December 2014

This document is set in Palatino, compiled with pdfLaTeX2e and `Biber`.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Statutory Declaration

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 marked all material which has been quoted either literally or by content from the used sources.

Graz, ____________________ ____________________

Date Signature

Abstract

English

Teamwork and collaboration skills are very important to improve the learning efficiency and experience. Teamsketch was developed to provide a collaborative sketch environment for iPads where pupils can draw one sketch together simultaneously. Up to four pupils take part in one session and train collaboration just by drawing a sketch. Features and issues of state-of-the-art applications were evaluated and solved by implementing a prototype from scratch in Apple's new programming language Swift.
Additionally, a web service as well as a web interface was implemented to provide an evaluation tool for teachers. Furthermore, pupils can upload and download their drawn sketches and profile pictures.
A first field test was carried out at the primary school Graz-Hirten which showed the potential of this app.

German

Kollaboratives Arbeiten und Teamfähigkeit sind wichtige Faktoren, um die Lernerfahrung und die Lerneffizienz zu verbessern. Teamsketch wurde als kollaborative Zeichen App für das iPad entwickelt, die das gleichzeitige Zeichnen von bis zu vier Personen an einer Zeichnung ermöglicht. Zuerst wurden die Funktionen und die Einschränkungen von aktuellen, kollaborativen Zeichen Apps analysiert. Anhand dieser Erkenntnisse wurde der Prototyp von Teamsketch entwickelt.
Zusätzlich zur App wurde ein Web-Service zur Verwaltung von Zeichnungen implementiert. Weiters ermöglicht eine Homepage für Lehrer/innen und Schüler/innen das Betrachten und Evaluieren von gemalten Bildern.
Abschließend wurde ein erster Feldversuch in einer Klasse der Volksschule Graz-Hirten durchgeführt, welcher das Potenzial der App zeigt.

Acknowledgements

I would like to extend my gratitude to my advisor Assoc. Prof. Martin Ebner who helped me whenever I had questions or needed support.

Furthermore, I would like to thank Silvana Aureli, who is the teacher of the iPad class Graz-Hirten, for allowing me to test the app with her pupils and gave me important feedback.

Additionally, I would like to thank Wolfgang Mesič and Hans-Peter Schlegl for providing their iPads more than once to test Teamsketch. Without them, fundamental testing would not have been possible.

Last but not least, I would like to thank my girlfriend Romana Marchel, who is in her last year of primary teacher's education, for supporting me during the whole design process as well as being there for me in the last few work-intensive months.

Contents

List of acronyms

API	Application Programming Interface
CSS	Cascading Style Sheets
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
ID	Identifier
IDE	Integrated Development Environment
IICM	Institute of Information Systems and Computer Media
MTOM	Message Transmission Optimization Mechanism
MVC	Model View Controller
PC	Personal Computer
PDF	Portable Document Format
PHP	PHP: Hypertext Preprocessor
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
SOC	Service Oriented Computing
SSH	Secure SHell
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTF	Unicode Transformation Format
WSDL	Web Services Description Language
WWDC	Apple Worldwide Developers Conference
XML	Extensible Markup Language

List of Figures

Listings

List of Tables

1 Introduction

Nowadays our children grow up with mobile devices. Some are able to use a cellphone/tablet even before they are able to write or read. In the United States 52% of children from 0-8 years have already used a mobile media platform such as smart phones and tablets [Rideout, 2013, p. 21] but only 28% of them have ever used educational game apps [Rideout, 2013, p. 23].
Even at older age (pupils at lower secondary school in Austria, age: 10-14) only 6% of the pupils use their device frequently for learning [Grimus and Ebner, 2014, p. 1603]. As mobile media plays a significant role in our lives, it is in our responsibility to teach children how to use modern media and technology wisely. They should learn to use their devices to improve their skills.
To be successful in life and business, team work and the ability to learn from each other (collaborative learning) are important skills. Teamsketch not only tries to fulfill these requirements but also tries to motivate.

1.1 Research Goal

The goal of this project is to build a prototype drawing app. Pupils should be able to work on a sketch simultaneously and all participants should get display updates immediately. Furthermore, the collaborative learning capabilities of drawing together simultaneously should be observed in a first field study.

1.2 Overview

This master's thesis covers development of an iPad sketch app to enable students to draw simultaneously on one sketch. Additionally, a web service and a web interface were implemented to provide a sketch review platform. Pupils can upload their sketches and teachers can access them to give an adequate feedback. The thesis consists of the following chapters:

- Chapter two analyses the theoretical background. Collaborative learning will be introduced and explained in mobile device context.

- Chapter three summarizes feature and issue analysis of already implemented collaborative sketch apps on different platforms. Based on this feature evaluation and testing conclusions were made which were considered during the project planning phase.

- Chapter four describes app features and details as well as implementation and testing.

- Chapter five covers the web service features, environment, implementation and testing.

- Chapter six describes the web interface for students and teachers in detail.

- Chapter seven explains and analyses the field study at a primary school.

- Chapter eight gives a short project summary followed by technical aspects and issues. Furthermore, the project outcome is elaborated as well as further study opportunities.

2 Theoretical Background

This chapter elaborates on the theoretical aspects of this project. At first the term ”collaborative learning” is introduced, then several parameters are defined to use mobile devices for collaborative drawing.

2.1 Collaborative Learning

Collaborative learning in simple terms describes situations where a group of people tries to learn something together [Dillenbourg, 1999, p. 1].

Cooperative learning consists of five main elements:

- positive dependency
- individual responsibility
- mutual support
- application of social competences
- reflection of group processes

Positive dependency means that people in a group try to achieve a collective goal. When they are aware of this element, the development of individual responsibility will be benefited as well as the mutual support. Cooperative learning requires multiple social skills such as leadership, communication, discretionary and conflict resolution. Cooperative learning is more complex than other learning interactions because participants are simultaneously busy with the given task and the interaction between group members. After the cooperative learning task is finished, the group work should be reflected. Participants should discuss which actions were helpful while learning together as well as which actions obstructed the learning process. During the reflection, the individual contributions should be discussed as well as how they provide contribution to the group target. Reflections have many positive effects such as increased self confidence, higher social skills and a positive attitude to the processed group task [[D. W. Johnson and R. T. Johnson, 1989][D. W. Johnson, R. T. Johnson, and Holubec, 2002][D. W. Johnson, 2003][D. W. Johnson and R. T. Johnson, 2005] quoted from [D. W. Johnson and R. T. Johnson, 2008, p. 18-20]].

There are three different types of collaboration groups according to time of collaboration [D. W. Johnson, R. T. Johnson, and Holubec, 2002].:

- pupils work together for a short period of time (a few minutes to one hour)

- pupils work together in one group for a medium period of time (one hour to several weeks) to reach common learning objectives

- pupils work together in heterogeneous groups for a long period of time (half a year to several years)

Teamsketch targets groups which work together for a medium period of time and try to reach a collaborative goal: a finished sketch.
Teachers have to manage several aspects of this form of collaborative work [D. W. Johnson and R. T. Johnson, 2008, p. 18-20]:

- group role and group size
  According to previously defined learning objectives, group roles and group size are selected. In Teamsketch, one pupil acts as a leader and starts the sketch. The leader could have more responsibilities such as dividing the sketch space and assign it to the other pupils who joined the sketch.

- task and positive dependency
  The teacher should explain to Teamsketch users that they all have to draw one sketch. Since everybody can delete and overwrite the work of others, the pupils have to draw well-considered.

- observation and intervention
  When pupils have problems and need assistance, the teacher should give them support.

- assessment and evaluation
  Teachers should evaluate the team performance as well as manage the reflection of the group performance.

2.2 Collaborative Learning with Mobile Devices

Several studies attested that the usage of mobile devices improved the computer based collaborative learning significantly [Zurita and Nussbaum, 2004, p. 300-301][Danesh

et al., 2001, p. 394].
The small size of the devices enables pupils to sit very close to each other and they can communicate easily with each other. Many pupils are already experienced in using mobile devices with touch screens [Rideout, 2013]. Therefor it is easier for children to use a small device with touch screen than sitting in front of a computer.
The smaller screen on mobile devices could also be a drawback, because of the lack of space. This issue should be targeted with a very well-considered user interface. Only most import features and icons should be displayed [Danesh et al., 2001, p. 392].
A significant advantage of using tablets to perform collaborative drawing is that pupils have their own space (device) to draw. There is no physical restriction as there is when they draw a real sketch on a paper in a group. While drawing a real sketch together, only one pupil can draw at a certain place. They have to divide space if they want to draw simultaneously. Additionally it is very difficult to get an overview of the sketch when four pupils are working with their pens on one big sheet. These restrictions do not occur if pupils use their own tablets to draw. Everyone has their own device and draws without bothering other pupils physically.
Another significant advantage of using a tablet rather than using a PC[1] is that pupils could use a stylus for drawing on the touch screen.

[1]Personal Computer

3 State-of-the-art Collaborative Sketch Applications

Before development of a new collaborative sketch app could be started, an investigation of currently available applications must be performed. The following list contains all investigated applications. All apps were tested on two devices (iPad2 and iPad4) to verify their features. Each test app represents a certain group of applications. The first app is a web based collaborative sketch app (path based drawing) which needs an internet connection even for Wi-Fi collaboration. The second drawing app (pixel based drawing) can be used without an internet connection. The last app represents browser based collaborative sketch apps.

- Baiboard
- Whiteboard Lite
- Flockdraw

3.1 Baiboard

Baiboard[1] is a sketch app which provides a collaborative white board. The main features are:

- web collaboration (up to 40 collaborators)
- Wi-Fi collaboration (internet access required)
- sketch zoom
- undo function
- squared line guides
- shape drawing (lines, circles, predefined symbols)

[1] http://www.baiboard.com, last accessed: 15. Aug. 2014

- custom shape import from Dropbox[2], photo album and iTunes[3] Sharing
- text insertion
- eight pen colors (black, dark blue, blue, light blue, green, purple, red and yellow)
- four pen stroke widths (fat plus, fat, medium and thin)
- whiteboard export as PDF[4] and image with transparent or white background
- whiteboard sharing on Internet (URL[5]), Dropbox, Evernote[6], Facebook[7], Twitter[8] and Tumblr[9]
- image import
- print support
- live voice conferencing (beta) and text chat

Figure 3.1 shows a screen shot of the drawing view.

3.1.1 Conclusion

The drawing responsiveness is fast, drawing is very comfortable with this app. All lines and shapes are accessible as paths. The rubber acts as a path removal tool, only whole paths can be deleted. A significant drawback of this app is that users have to be connected to the internet in order to use real time collaboration.

[2]https://www.dropbox.com, last accessed: 30. Nov. 2014
[3]https://www.apple.com/itunes, last accessed: 30. Nov. 2014
[4]Portable Document Format
[5]Uniform Resource Locator
[6]https://evernote.com, last accessed: 30. Nov. 2014
[7]https://www.facebook.com, last accessed: 30. Nov. 2014
[8]https://twitter.com, last accessed: 30. Nov. 2014
[9]https://www.tumblr.com, last accessed: 30. Nov. 2014

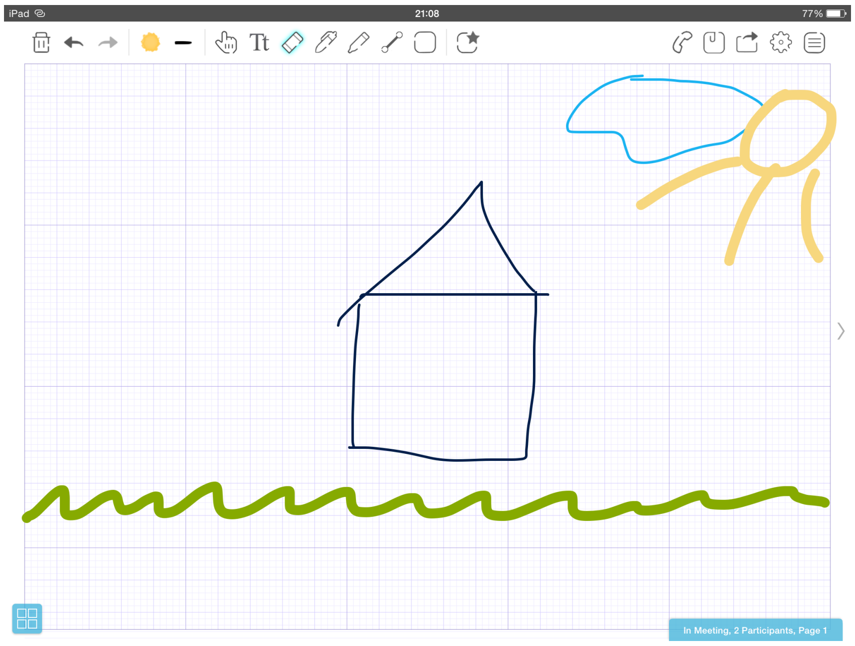


Figure 3.1: Baiboard draw screen - iPad

3.2 Whiteboard Lite

Whiteboard Lite[10] is available for iPad and supports collaborative drawing over web and Wi-Fi. The following list elaborates on the main features of this app.

- web collaboration
- collaboration over Wi-Fi
- auto save
- import images from camera roll
- import text
- imported items are movable
- ten predefined colors
- custom colors available by selecting color in color palette.
- color alpha channel (opacity)

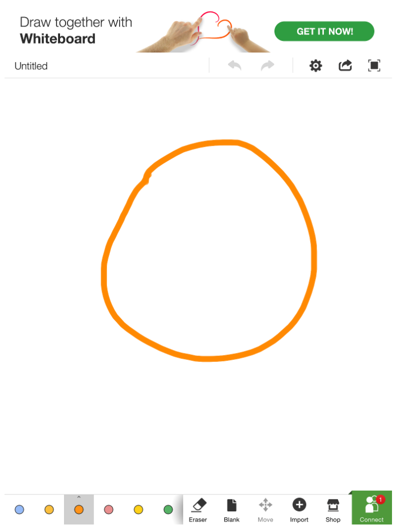
[10]http://www.greengar.com, last accessed: 19. Nov. 2014

- variable stroke width (width slider)
- video out
- sketch sharing on Facebook, Twitter, iCloud[11], E-mail, Google Drive[12], Evernote and Whiteboard gallery
- sketch zoom
- undo function
- API[13] available
- full screen mode

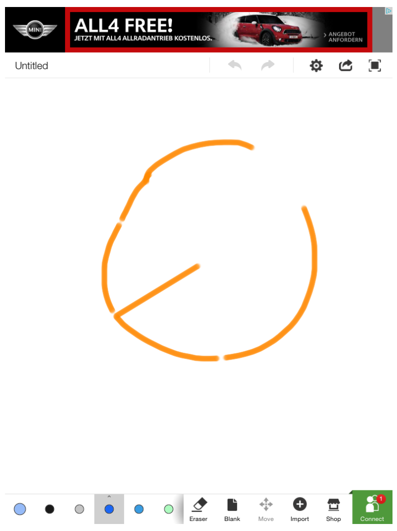
3.2.1 Conclusion

This app uses pixel based drawing. Real-time drawing over Wi-Fi works without requiring an internet connection but sketch synchronization is sometimes delayed and sketches contains wrong transmitted strokes, Figure 3.2 shows drawing screen details and synchronizing differences after drawing a circle. The strokes look very cornered when the user often changes the drawing direction (Figure 3.3). This may be caused by the lack of a curve-fitting and line smoothing algorithm.

[11] `https://www.icloud.com`, last accessed: 30. Nov. 2014
[12] `https://www.google.com/drive`, last accessed: 30. Nov. 2014
[13] Application Programming Interface



(a) 1st user



(b) 2nd user - transmitted shape

Figure 3.2: Whiteboard Lite synchronization errors

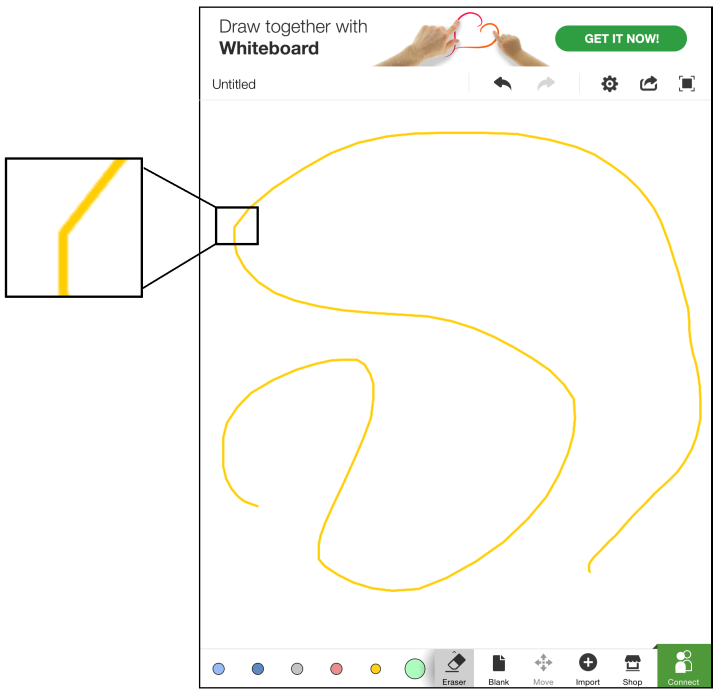


Figure 3.3: Whiteboard Lite line shape

3.3 Flockdraw

The Flockdraw[14] app for iPad is only available in the Apple Store of the United States therefor the iPad app could not be tested. Only the web based version was tested. Flockdraw provides the following main features:

- text input
- area color fill
- line drawing
- storing sketches online
- sharing sketches on Facebook and Twitter
- six stroke widths
- full color palette
- text chat

3.3.1 Conclusion

Flockdraw web application needs the Flash player[15] therefor the application cannot be used on the iPad as there is no native Flash support available [Jobs, 2010]. When using Flockdraw within a browser on a PC, real-time collaborative drawing works by joining an active session with a session ID[16]. Drawing long and curved strokes lead to cornered shapes. Stroke synchronizing performance highly depends on internet speed and latency. Synchronization errors occurred from time to time. Figure 3.4 and 3.5 show draw screen shots with two involved users in one active drawing session.

[14] http://flockdraw.com, last accessed: 17. Aug. 2014
[15] http://get.adobe.com/de/flashplayer, last accessed: 17. Aug. 2014
[16] Identifier

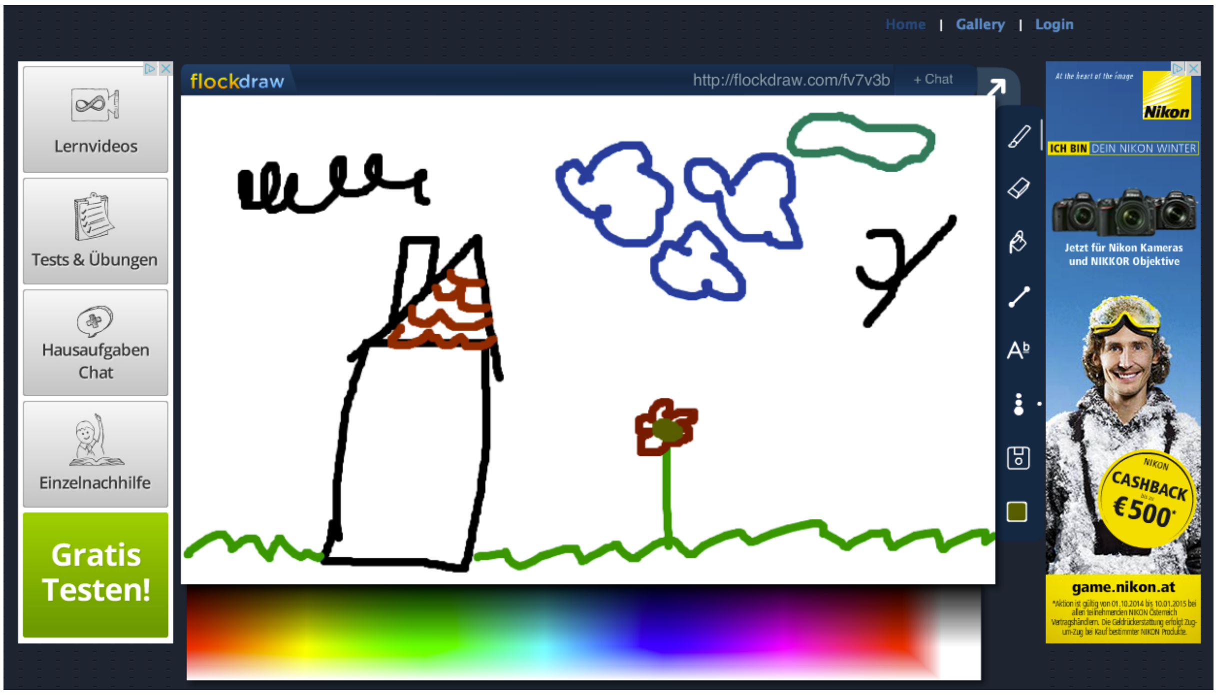


Figure 3.4: Flockdraw screen session - first user

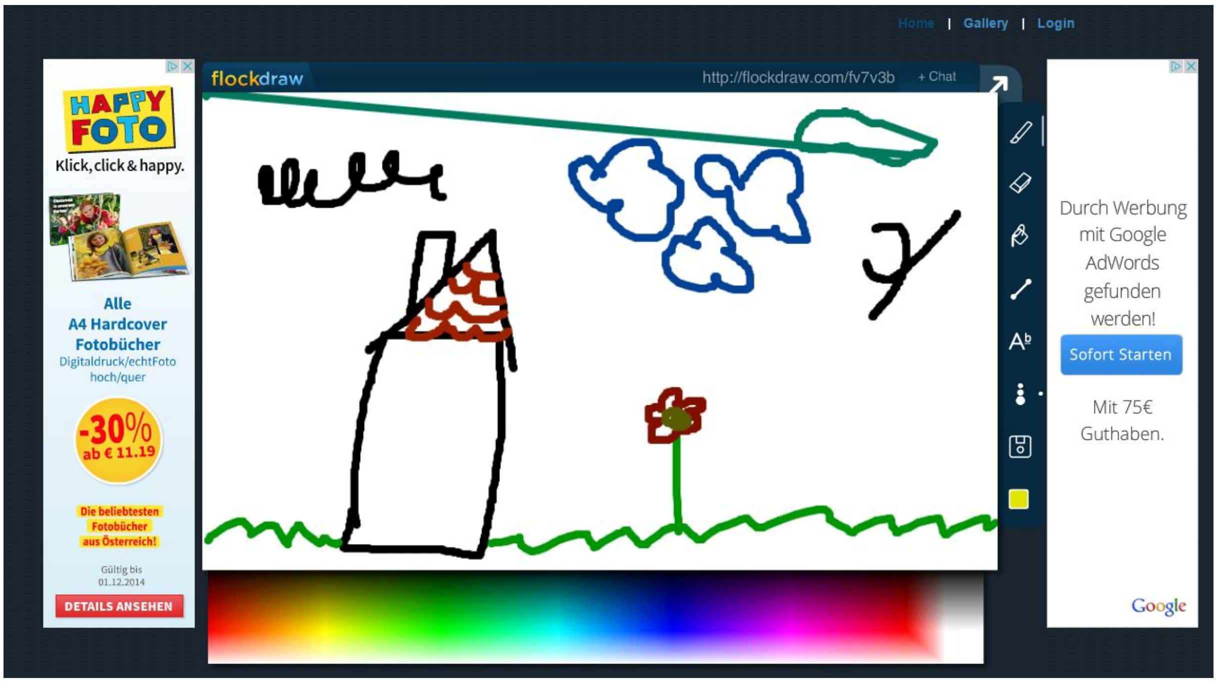

Home | Gallery | Login
flockdraw
http://flockdraw.com/fv7v3b
+ Chat
HAPPY FOTO
Klick, click & happy.
Alle A4 Hardcover Fotobücher
Digitaldruck/echtFoto hoch/quer
-30% ab € 11.19
Die beliebtesten Fotobücher aus Österreich!
Gültig bis 01.12.2014
DETAILS ANSEHEN
Durch Werbung mit Google AdWords gefunden werden!
Sofort Starten
Mit 75€ Guthaben.
Google

Figure 3.5: Flockdraw screen session - second user

Table 3.1: Feature comparison - app features

Application	Web collaboration	Wi-Fi collaboration	printing	available on iPad
Baiboard	●	● (internet connection required)	●	●
Whiteboard Lite	●	●		●
Flockdraw	●		●	iPad App in US App store (not tested)

Table 3.2: Feature comparison - drawing features

Application	colors	stroke widths	zoom	line drawing	text	undo function	flood fill	predefined shapes	user defined shapes	shapes movable
Baiboard	8	4	●	●	●	●		●	●	●
Whiteboard Lite	full palette	slider	●	●	●	●		●	●	●
Flockdraw	full palette	6		●	●		●			

3.4 Feature comparison

Table 3.1 shows general feature comparison of all tested applications. Table 3.2 elaborates on drawing related features, table 3.3 focus on data operations such as import, export and saving. Table 3.4 lists additional features.

Table 3.3: Feature comparison - import / export / save

Application	import sketches	export sketches	import predefined shapes	import user defined shapes	save sketches
Baiboard	pictures, pdf and maps	web (URL), pdf, photo album, iMessage, Email, Dropbox, Evernote, Facebook, Twitter and Tumblr	rectangle, circle, speech balloon, bubbles, dancing men, farsi chehre, gesture, misc., mockups, network topology	user defined library with import from photo album, iTunes sharing and Dropbox	offline board, snapshots in Baiboard cloud and as image to camera roll
Whiteboard Lite		Facebook, Twitter, iCloud, Whiteboard Gallery Email, Google Drive, Evernote	stickers	camera and photos	auto save as image to camera roll
Flockdraw		Flockdraw web page, Facebook and Twitter			

Table 3.4: Feature comparison - additional features

Application	chat	API	full screen
Baiboard	voice chat (beta) and text chat		
Whiteboard Lite		●	●
Flockdraw	text chat		●

3.5 Conclusion

After investigation of these three collaborative drawing applications, the following conclusions could be made:

- Real-time sketch collaboration is already implemented in some applications but suffers from draw update latency and/or synchronization errors. Fixing this issue is declared as one of the main targets in this research project as well as implementing the application without requiring an internet connection. Since the main area of application of Teamsketch are school classes, requiring an internet connection would lead to high latency because all pupils would use the same internet connection at once. To address this issue Bluetooth should be available in case of Wi-Fi is not working or the performance is not satisfying.

- In Teamsketch only basic drawing tools will be provided, no layers, text or customized shapes. Instead of drawing, the main focus should be on collaboration and not on multiple app features.

- The undo function will not be implemented because the app should encourage efficient and well-considered drawing. This assumption will be investigated during the prototype test in a primary school class.

- No chat function will be added because pupils will sit on one desk together while they are drawing and should communicate in person.

- No predefined shapes (circles, polygons and letters) will be implemented because students should practice free hand drawing to improve their creativity and fine motor skills.

- Drawn strokes should be optimized to look more realistic (less cornered) when user often change the draw direction (no sharp directional changes) therefor a line smoothing and curve fitting algorithm should be used.

4 Teamsketch App

Teamsketch is an app for iPad which supports collaborative drawing of a single sketch with up to four devices.

4.1 Idea

There are already apps available with collaborative drawing support but many of them have issues with responsiveness and accuracy. Furthermore, they often require an internet connection. This project addresses these issues because pupils should not be constrained by technical limitations as drawing lags and synchronizing problems or a mandatory internet connection. The aim of this app is to increase team skills by drawing a picture. Up to four pupils take part in one session and draw together. They should be encouraged to discuss their drawing and who is responsible for which part of the sketch.

4.2 Development Environment

Teamsketch is an iOS app written in Swift. Apple's new programming language replaces Objective-C as main development language. Swift was introduced at the WWDC[1] in June 2014 [Isted and Addey, 2014]. Apple provides an IDE[2] named xCode with all necessary features to develop and test iOS apps written in Swift. Teamsketch app development started shortly after the WWDC with xCode 6 Beta 3.

4.3 Features

Teamsketch should help children to improve their team skills and should support collaborative learning. Therefore various requirements were defined:

- 2-4 pupils should have the possibility to draw a sketch simultaneously

[1]Apple Worldwide Developers Conference
[2]Integrated Development Environment

- real-time drawing and screen updating
- stroke smoothing and curve fitting algorithm to prevent cornered, unnatural stroke shape
- sketch zoom and scrolling
- sketch position overview while zooming and scrolling
- twelve available colors for drawing (white, yellow, orange, red, pink, purple, light green, dark green, light blue, dark blue, brown and black)
- optional line guides
- two different felt pen stroke widths
- user management
- no internet connection required for drawing
- Wi-Fi and Bluetooth support for peer connections
- pupils should be able to invite and decline others to their sketch
- custom profile picture creation
- upload/download sketches to a web service
- sketch storing on device

4.4 System Architecture

Teamsketch is built with xCode 6.1 (6A1052d) using iOS SDK[3] 8.1. Supported iPad devices are iPad 2, iPad 3, iPad 4, iPad Air and iPad Air 2 as well as iPad Mini, iPad Mini 2 and iPad Mini 3. The reason why iOS SDK 8.1 was chosen is that tests showed that multi-peer drawing performance was highly dependent on the used iOS version. Only supporting the newest iOS version is not a big issue because Apple users usually update their devices shortly after release. Apple states the iOS 8 adoption rate as 56% on November 10, 2014[4], release of iOS 8.1 was October 20, 2014[5].

[3] Software Development Kit
[4] `https://developer.apple.com/support/appstore/`, last accessed: 19. Nov. 2014
[5] `https://developer.apple.com/devcenter/ios/index.action`, last accessed: 19. Nov. 2014

Teamsketch was implemented for iPads because of two main reasons:

- there is an iPad school class available in primary school Graz-Hirten which allows fundamental testing of the app in real world environment.
- the University of Technology Graz, especially the group Social Learning of the institute IICM[6], has several years of experience in implementing iOS learning applications for iPads and testing them in school classes. This project is a continuation research based on the experience from previous projects. It benefited much from already existent infrastructure and knowledge. Several projects such as Buchstabenpost[7] already encountered collaborative learning in multi-peer environment.

4.4.1 Frameworks

Teamsketch uses Apple's Foundation, UIKit, CoreGraphics and the MultipeerConnectivity framework.

Foundation framework

Foundation framework[8] provides classes for data storage, timing, URL session management and other essential basic functionality.

UIKit framework

UIKit framework[9] is a high level framework for managing views, view controllers, motion events, drawing as well as other important app features. UIKit implicitly uses Core Animation framework to provide view animation. Basic drawing functionality is provided by the lower Core Graphics framework.

[6]Institute of Information Systems and Computer Media

[7]https://itunes.apple.com/at/app/buchstaben-post/id736836885?mt=8, last accessed: 08. Nov. 2014

[8]https://developer.apple.com/library/ios/documentation/cocoa/reference/foundation/objc_classic/index.html, last accessed: 06. Nov. 2014

[9]https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIKit_Framework/, last accessed: 06. Nov. 2014

Core Graphics framework

Core Graphics framework is an API implemented in C. Path-drawing, transformations, color management, off-screen rendering and other basic image tasks can be realized with Core Graphics. Many tasks can also be implemented with UIKit in a higher level programming context.

Multipeer Connectivity framework

The Multipeer Connectivity framework implements services for nearby devices, it supports Wi-Fi infrastructure and ad hoc networks as well as Bluetooth connections. The framework manages low level networking and communication. Data can be exchanged by messages, streams and resource data.

4.4.2 Basic Concepts

This chapter elaborates on basic concepts which were used within the Teamsketch app.

Drawing in iOS

The iOS drawing stack[10] as shown in figure 4.1 provides drawing functionality in different abstraction levels.

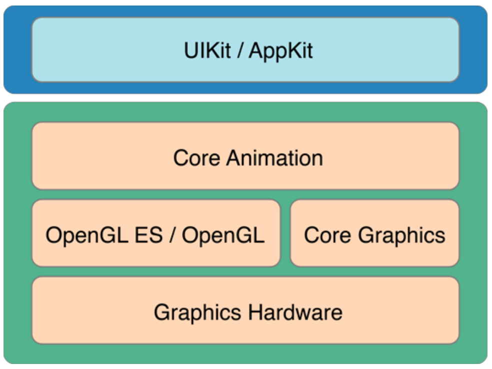


Figure 4.1: iOS drawing stack

[10] https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/CoreAnimation_guide/Introduction/Introduction.html, last accessed: 06. Nov. 2014

Teamsketch app uses a customized view (UIView) for drawing. A UIView is redrawn under the following circumstances[11]

- a part of a view or the view itself became visible
- the "hidden" property was set to "false"
- scrolling a view out off the screen and then back
- redrawing can be triggered programmatically by calling "setNeedsDisplay" method of the view

iOS supports on screen drawing as well as off-screen rendering. To perform a drawing operation a graphics context is needed. After drawing this context has to be closed.

Multipeer Connections

The Multipeer Connectivity framework[12] was introduced in iOS 7.0. A multi-peer connection is managed by several components. Teamsketch uses the following components to establish a multi-peer connection:

- Session Objects (MCSession) hold the connected peers and manage connection and disconnection of peers. New peers can create a session or join an active session.
- Advertiser (MCAdvertiserAssistant) notifies other reachable peers that the app wants to join an active session and provides a basic user interface to accept invitations.
- Browser (MCNearbyServiceBrowser) searches for advertising peers nearby.
- Peer IDs (MCPeerID) are unique identifiers assigned to each peer.

Web service communication

Teamsketch uses HTTP[13] POST requests within a NSURLSession to communicate with the web service. The app uses a data task to send and receive data (NSData) objects.

[11] https://developer.apple.com/library/ios/documentation/2DDrawing/Conceptual/DrawingPrintingiOS/GraphicsDrawingOverview/GraphicsDrawingOverview.html, last accessed: 06. Nov. 2014

[12] https://developer.apple.com/library/IOs/documentation/MultipeerConnectivity/Reference/MultipeerConnectivityFramework/index.html, last accessed: 06. Nov. 2014

[13] HyperText Transfer Protocol

4.5 Swift

Teamsketch is implemented in Swift. Apple's new programming language introduces new programming techniques which simplifies iOS app development significantly. The following swift features and program techniques[14] were used (some examples):

- optionals
  In Swift optionals are used for variables which can be absent. This concept is similar to returning nil in Objective-C. This worked only for objects but not for structures, enumerations as well as basic C types. In swift optionals can be used for any type at all. Listing 4.2 shows usage of optionals as condition for a while loop. This code fragment is used to count the number of elements in an enumeration with following constrains:
  - enumeration is an integer type
  - raw value of elements start at zero and increases

  The "rawValue" constructor returns an optional. If there is no element with the specified rawValue, the optional holds "empty value". A constant, assigned with an optional, can be coerced to "true" if optional holds a value and to "false" if optional holds "empty-value" therefore this assignation can be used as a loop condition. These features decrease the needed lines of code significantly.

```
var numberOfElements = 0
while let currentGameMode = GameMode(rawValue: numberOfElements){
 numberOfElements++
}
```

Listing 4.1: Optionals as condition in loops

- type safety
  Swift is type safe. Therefore only variables and values of the same type can be assigned.

- extensions
  Extensions can be used to add functionality to already implemented classes. Teamsketch uses extensions to implement protocols. Using extensions tidies up class methods because functions, which belongs to a protocol, can be grouped easily.

- closures
  Closures are code blocks which can define input and output parameters. These code blocks can be used in various locations within program code. Listing 4.2

[14]https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html, last accessed: 06. Nov. 2014

shows closure usage to custom sort an image object array (smallSketches) according to the image property: number.

```
smallSketches.sort({$0.number < $1.number})
```

Listing 4.2: Image sort in array

4.6 Design Patterns

A design pattern, not necessarily in programming context, describes a problem and one approach to solve it and consequences which are the side effects of solving the particular problem. In many cases there is more than one solution of the same problem but with different trade-offs, complexity or usability [Gamma et al., 1995, p. 3].

4.6.1 Model/View/Controller

MVC[15] unions many design patterns into one architecture. It separates a program into three different parts [Gamma et al., 1995, p. 5]:

- Model - the application object which holds the data.
- View - screen representation of program data
- Controller - manages user input and other control logic

In iOS development context the model is implemented in separated model classes. A view controller (UIViewController) implements the connection between views and the model. Views can be implemented in two different ways: They can be added in storyboard to the view controller's parent view, or they can be added programmatically in the view controller class.

4.6.2 Singleton

A singleton is a class which has only one instance. This instance can be accessed from every location in program code and acts as a global [Gamma et al., 1995, p. 127]. In Teamsketch only one user is logged in at a time. The UserInformation singleton holds user name, id, profile pictures and user sketch thumbnails. This singleton combines all user related data into one global object.

[15]Model View Controller

4.6.3 Delegate

The delegate pattern describes forwarding of operations to other components [Gamma et al., 1995, p. 20]. Delegation is very important in iOS development. One object can move operations/methods to another object. In Teamsketch the session object retrieves the data from network and delegates the received data to other classes to perform adequate operations. The session delegates the messages according to its purpose. Sketch related messages are delegated to the sketch view, other game related information are delegated to the responsible view controller.

4.6.4 Observer

This pattern describes the forwarding of object updates to its dependencies [Gamma et al., 1995, p. 293]. In Swift the observer pattern can be easily implemented for class properties. Each property has "willSet" and "didSet" methods which notify other program parts that their value has changed or will change. Other supported ways to implement the Observer pattern are using notifications or Key-Value-Observing[16].

4.6.5 Facade

The goal of the facade pattern is to hide lower level functions and combine them to one simple high level API [Gamma et al., 1995, p. 185]. In Teamsketch the web API is hidden in TeamsketchSOAP class. This class manages all HTTP URL connections as well as the XML[17] parsing and request construction. Figure 4.2 shows the details of the facade pattern used within Teamsketch. Other classes just call high level functions to set or get data from the web service.

[16] `https://developer.apple.com/library/ios/documentation/swift/conceptual/Swift_Programming_Language/Properties.html`, last accessed: 20. Nov. 2014

[17] Extensible Markup Language

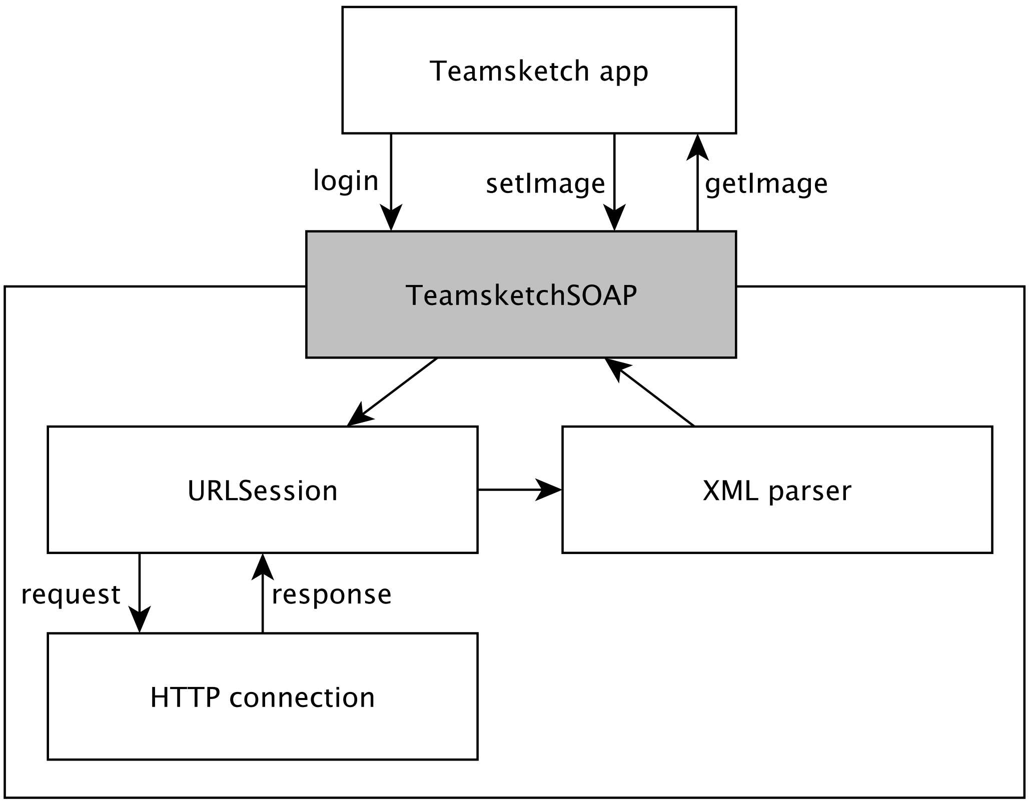


Figure 4.2: Facade pattern in Teamsketch

4.7 User Interface

This section gives an overview of the UI[18], its transitions and view details. Figure 4.3 shows all involved view controller segues.

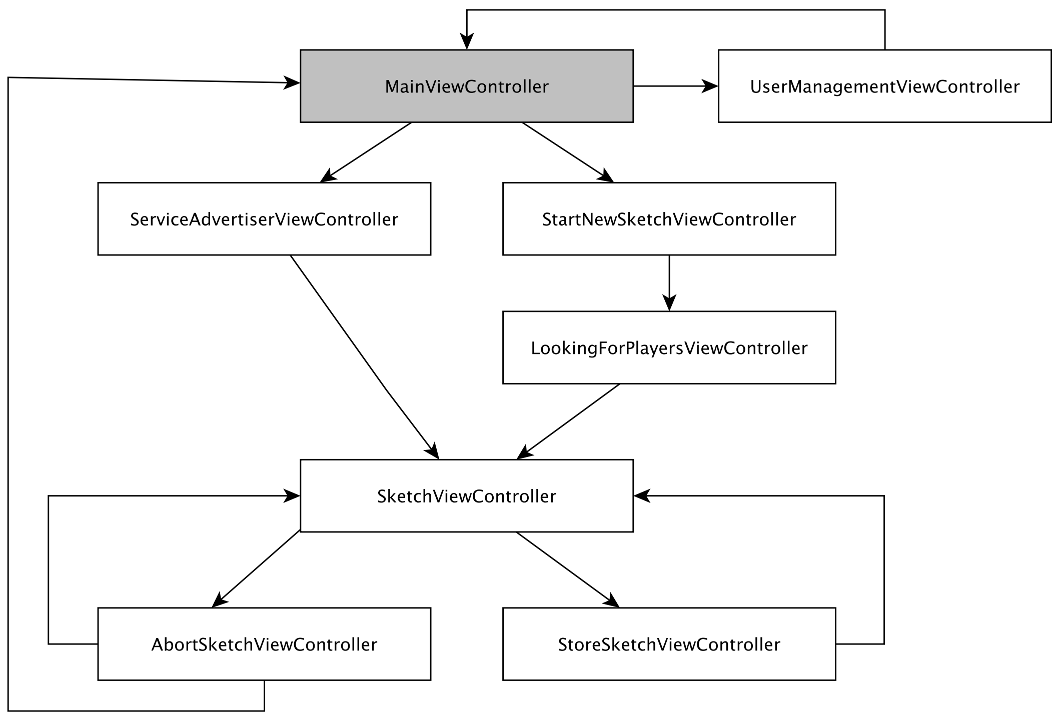

MainViewController
UserManagementViewController
ServiceAdvertiserViewController
StartNewSketchViewController
LookingForPlayersViewController
SketchViewController
AbortSketchViewController
StoreSketchViewController

Figure 4.3: Teamsketch UI screen transitions

The app starts with the MainViewController. Players can start a new sketch, join an active sketch or log into the web service. When players decide to start a new sketch, the StartNewSketchViewController will be shown. In this controller custom game parameters can be set. After confirming the game parameters the LookingForPlayersViewController appears where other reachable peers are listed. These peers can be invited to the sketch and already invited peers can be declined. When at least two players joined the session (one sketch creator plus at least one client), the game host can start the sketch and the SketchViewController will be shown. When players select "join sketch" on the main screen, the ServiceAdvertiserViewController will appear. In this screen, players advertise themselves over Wi-Fi or Bluetooth and game hosts can invite them to join a session. After joining the app waits until the game hosts starts the sketch. After receiving the start command the SketchViewController will be shown. This screen is the main game screen where all joined peers can draw one sketch simultaneously. Within this screen, two pop-up controllers are available: StoreSketchViewController supports storing the sketch online or on the device, AbortSketchViewController aborts

[18] User Interface

the sketch or restarts the game with an empty sheet. All user interactions, including transitions caused by other peers, are described in Figure 4.4.

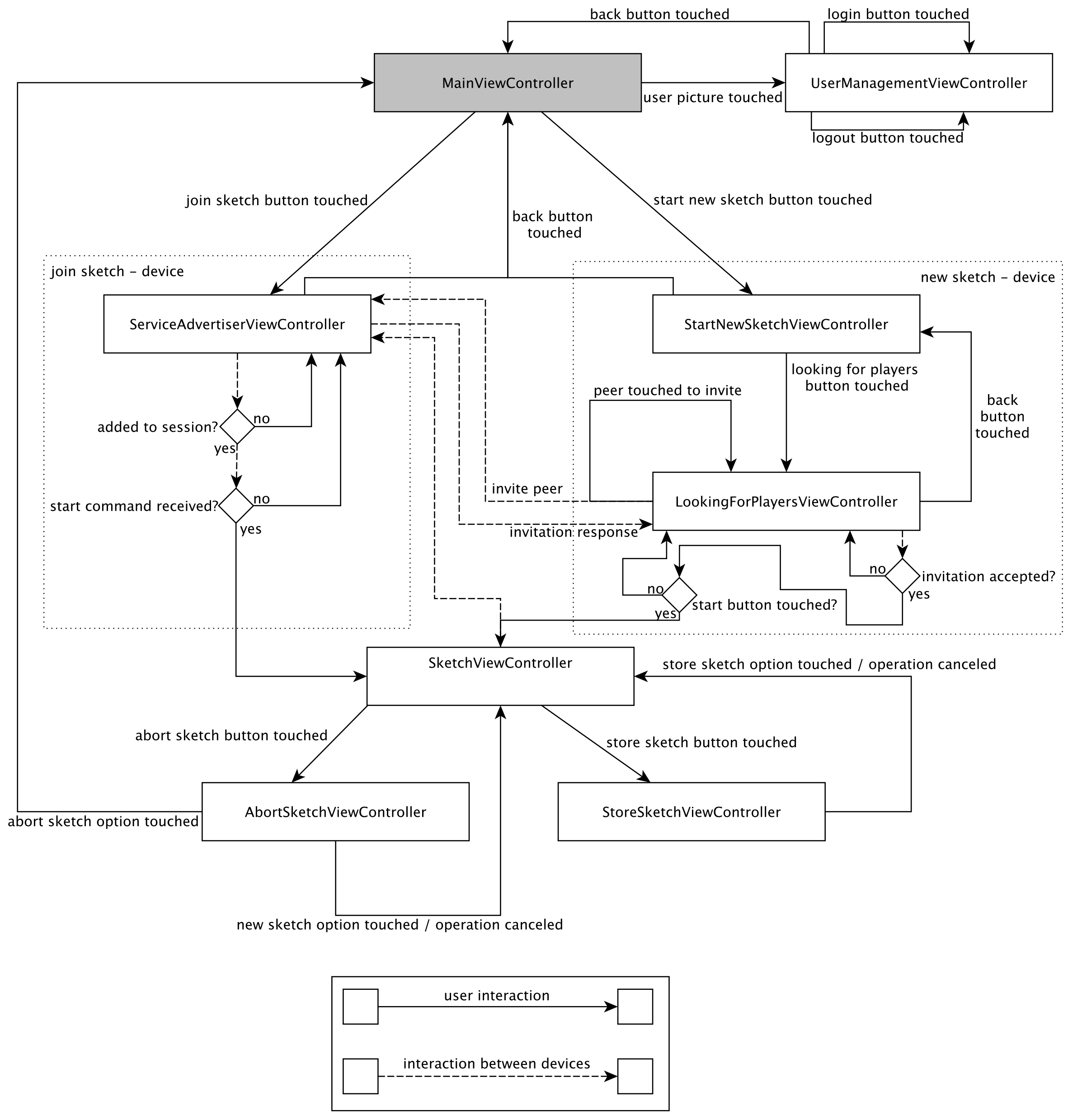
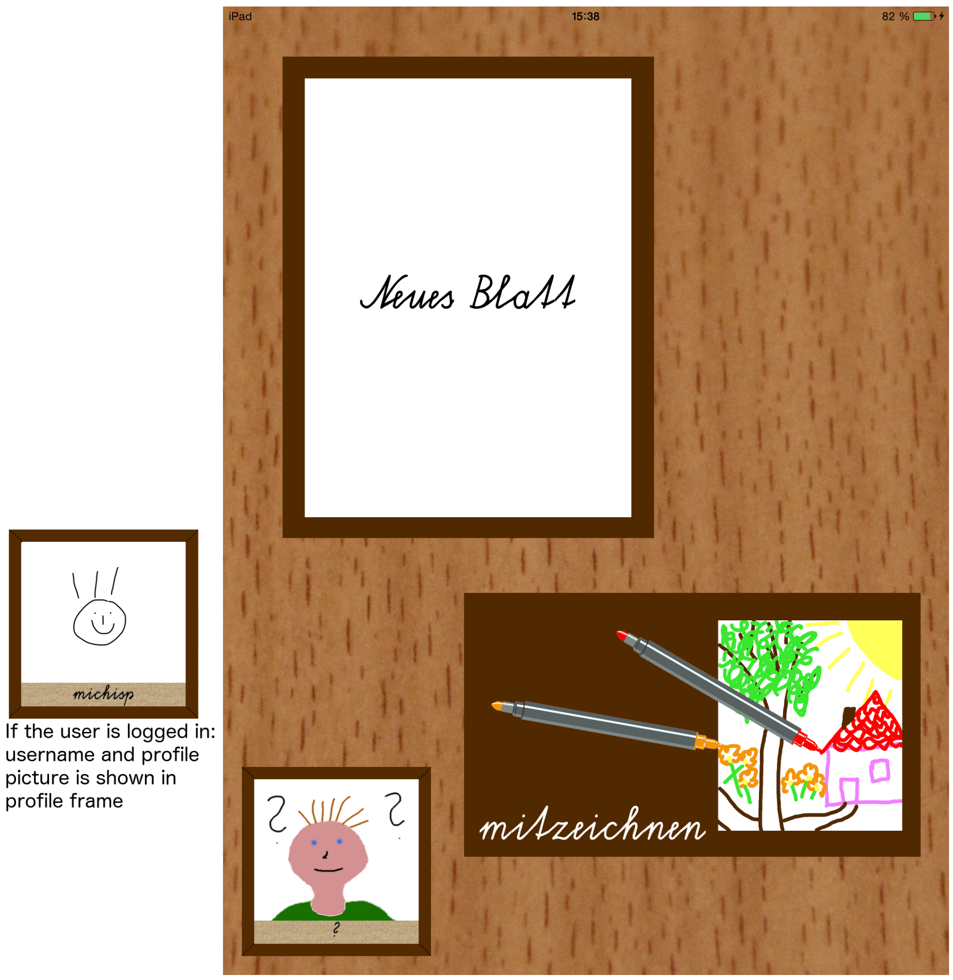

back button touched
login button touched
MainViewController
user picture touched
UserManagementViewController
logout button touched
join sketch button touched
back button touched
start new sketch button touched
join sketch – device
new sketch – device
ServiceAdvertiserViewController
StartNewSketchViewController
added to session?
no
yes
looking for players button touched
peer touched to invite
back button touched
start command received?
no
yes
invite peer
LookingForPlayersViewController
invitation response
no
invitation accepted?
yes
no
start button touched?
yes
SketchViewController
store sketch option touched / operation canceled
abort sketch button touched
store sketch button touched
abort sketch option touched
AbortSketchViewController
StoreSketchViewController
new sketch option touched / operation canceled
user interaction
interaction between devices

Figure 4.4: Teamsketch UI flowchart

4.7.1 Main Screen View Controller Scene

After starting the app, the splash screen is shown followed by the main screen view controller scene. Figure 4.5 shows the screen details. Three buttons can be touched:

- new sketch button (top, left)
- join sketch (center, right)
- user management (bottom, left)



If the user is logged in: username and profile picture is shown in profile frame

Figure 4.5: Main View Controller Scene

4.7.2 User Management View Controller Scene

After touching the profile picture frame in Main Screen View Controller, the User Management View Controller scene appears. There are two different states:

- user is logged out
- user is logged in

If the user is not logged in, the following views (Figure 4.6) are visible:

- user name text field
- password text field
- login button

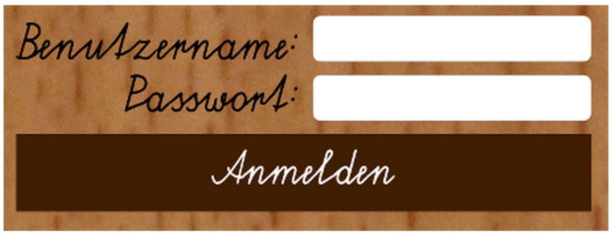
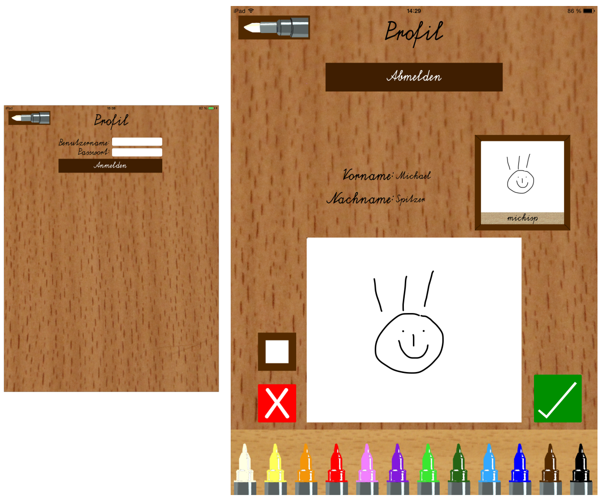


Figure 4.6: User Management View Controller - login sub view

If the user is logged in, an additional sub view is shown with profile details such as user's first and last name as well as the profile pic thumbnail and a sketch area where users can edit their profile picture. Sketch related functionality such as felt pens in color bar are elaborated later in the Sketch View Controller Scene (section: 4.7.6). Users can perform the following actions while they are logged in:

- logout
- revert profile picture to version stored online
- clear profile picture
- store profile picture online

Figure 4.7 shows the screen details of both states.



Vorname: Michael
Nachname: Spitzer

Figure 4.7: User Management View Controller

4.7.3 Start New Sketch View Controller Scene

Users, who start a new sketch, can set various game parameters such as:

- player name (optional, only visible, if the user is not logged in)
- sketch name
- time limit. This disables drawing functionality after a defined time interval (slider) between zero and 120 minutes. Users can store their sketches even after time has run out.
- line guides. Selectable guides:
  - no guides
  - squared
  - lined

If the user is logged in, the player name view group will be hidden. Figure 4.8 shows the scene details.

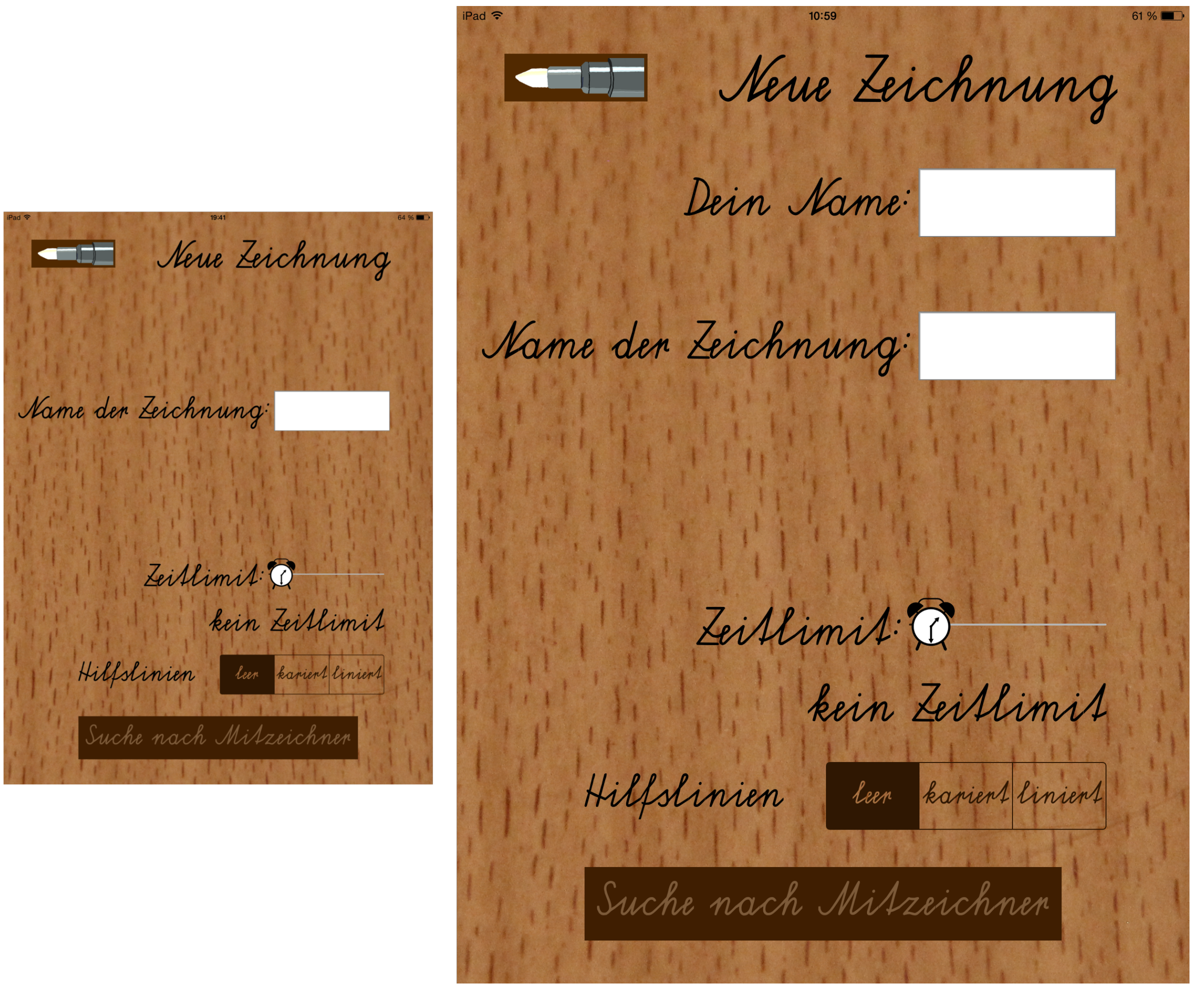


Figure 4.8: Start New Sketch View Controller Scene

4.7.4 Looking For Players View Controller Scene

After setting sketch related parameters, the LookingForPlayersViewController scene is shown. The upper table view lists all available devices (devices which are running Teamsketch in "join sketch" mode) nearby. When the user touches a peer in the list, an invitation is sent to the peer. If the peer accepts the invitation, the peer's name appears in the table view above. The host can decline an already accepted invitation by sliding left the corresponding peer name in the lower table view. When at least 2 players are in a session (one host and 1-3 peers), the "start sketch" button becomes enabled and the host can start the sketch. Figure 4.9 shows a usual scene flow.

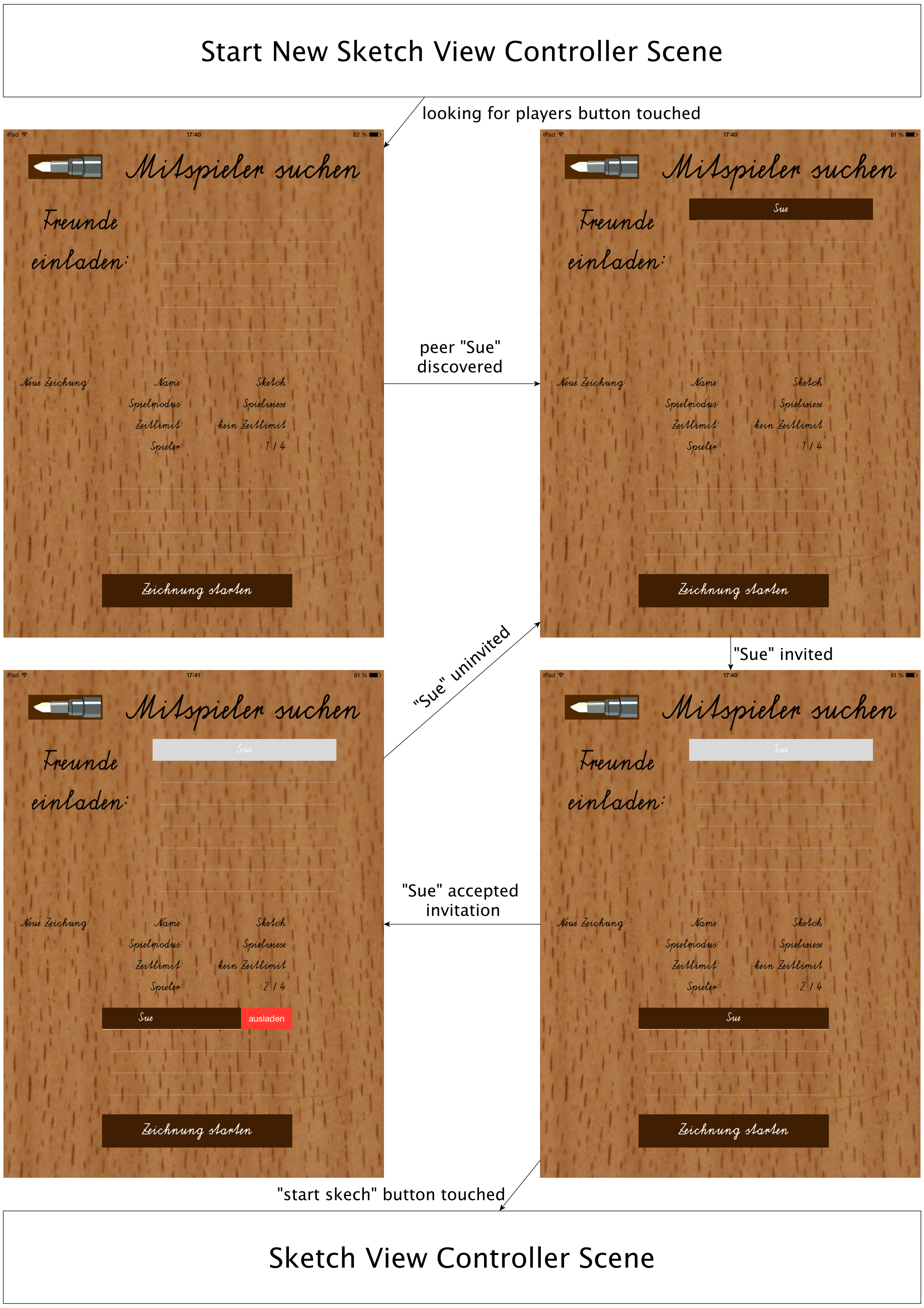


Figure 4.9: Looking For Players View Controller flow

4.7.5 Service Advertiser View Controller Scene

If the user has selected "join sketch" on the main screen, the Service Advertiser View Controller appears. If user is logged in, the advertising process starts immediately. Not authenticated users have to enter a player name and start the advertising process by touching the "looking for sketches" button. When the device is in advertising mode, other players who already started the "new sketch" process (game host) can see the device in the nearby peers list. After the host sent an invitation, a confirmation alert is shown on the advertising device. After the invitation is confirmed, game details will be shown in the bottom of the screen. The advertising peer stays in this state until the host declines the invitation or starts the sketch. Figure 4.10 shows a usual flow for not authenticated users.

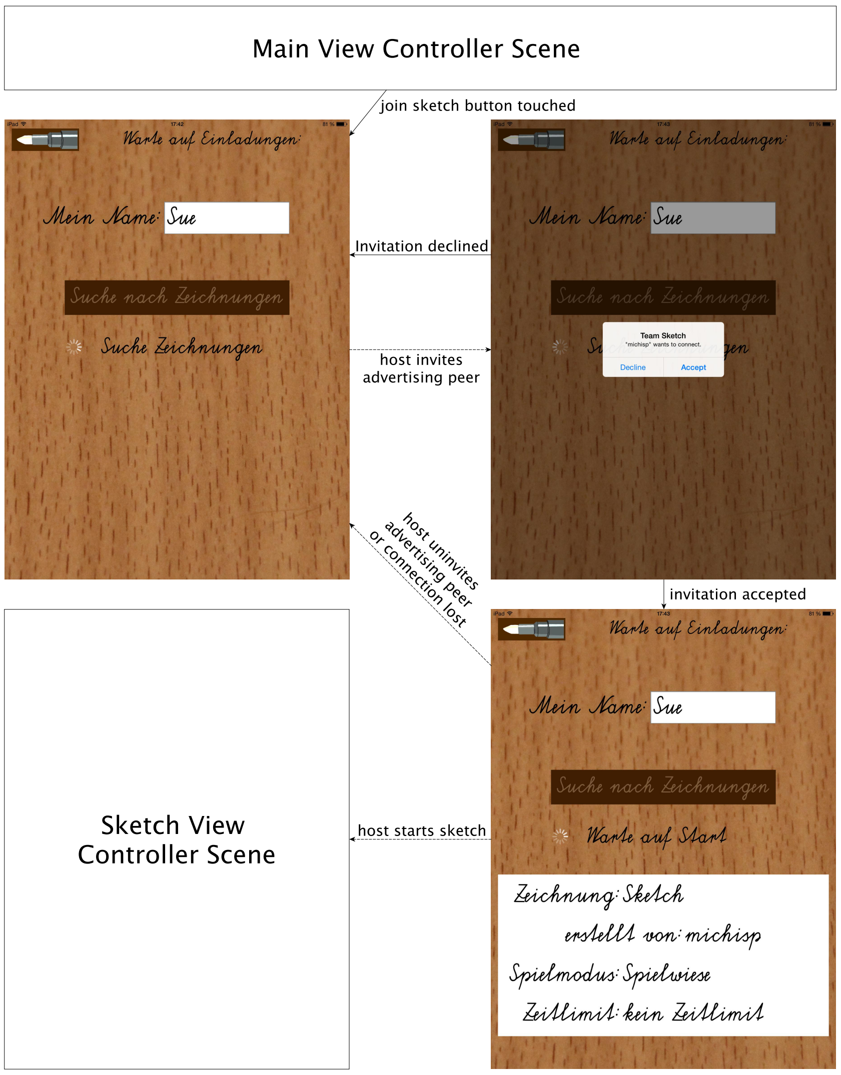


Figure 4.10: Service Advertiser View Controller flow

4.7.6 Sketch View Controller Scene

In this view all drawing actions take place. Users can select the desired color and stroke width in the color bar at the bottom of the screen. The red "X" on the left opens the Abort Sketch View Pop-up Controller and the green tick on the right opens the Store Sketch View Pop-up Controller. Figure 4.11 shows the details of the Sketch View Controller Scene.

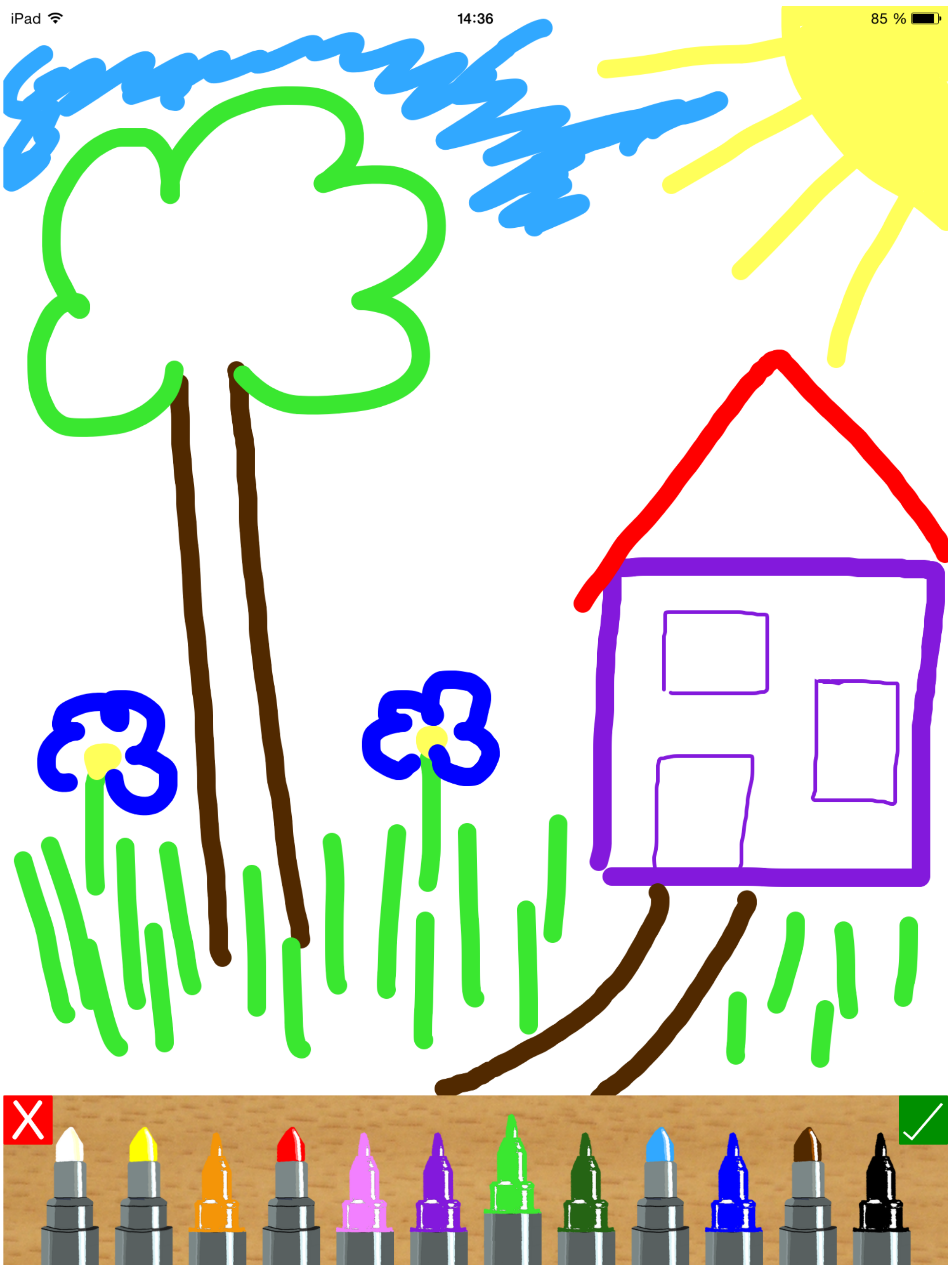


Figure 4.11: Sketch View Controller Scene

Felt Pen Color Bar

The drawing toolbox consists of twelve felt pens. With a single tap the desired color becomes active. When the user double taps on a felt pen, the felt pen becomes active and toggles its stroke width. To make it obvious which pen is active, the felt pen will grow. Figure 4.12 shows details of the felt pen switching process.

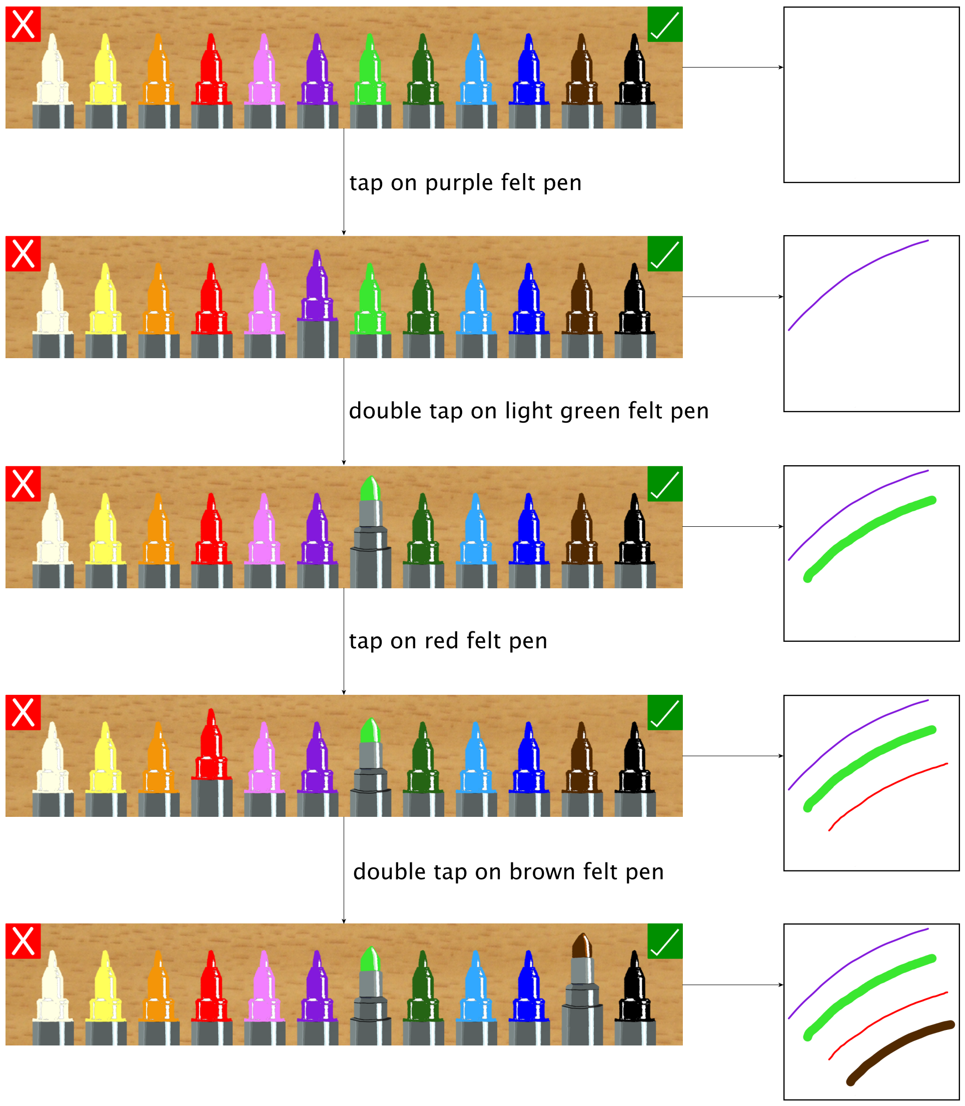

tap on purple felt pen
double tap on light green felt pen
tap on red felt pen
double tap on brown felt pen

Figure 4.12: Felt Pen Switching Process

Abort Sketch View Pop-up Controller

This controller is included into the Sketch View Controller as a UIPopUpController. Users can abort drawing or restart the sketch with all drawing cleared (Figure 4.13).

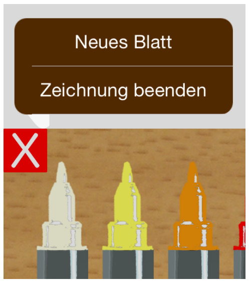
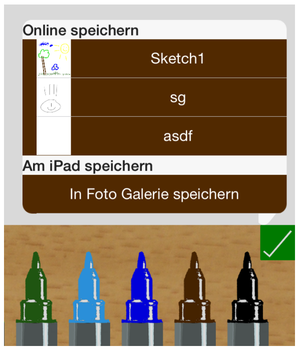
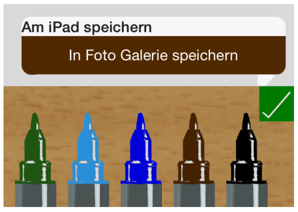


Figure 4.13: Abort Sketch View Pop-up Controller



(a) user is logged in



(b) user is logged out

Figure 4.14: Store Sketch View Pop-up Controller

Store Sketch View Pop-up Controller

The Store Sketch View Controller supports saving the sketch. If the user is logged in, sketches can be stored online, otherwise only the ”store on device” option is visible (Figure 4.14).

4.8 Implementation details

This chapter elaborates on the implementation details. Components with gray background color are standard components already included in the iOS SDK hence no explanation or detailed description is provided.
Attributes and methods in class diagrams without access modifier are "internal" meaning that they are accessible in any source file within a module[19].

4.8.1 Overview

All involved components are shown in figure 4.15.

4.8.2 Delegates

The delegation pattern in Swift is implemented within protocols. These components only include function descriptions which have to be implemented by the class which uses this delegate. No stored properties are allowed. Some of the functions can be declared "optional" therefore no implementation is necessary[20].

AbortSketchViewControllerDelegate

The AbortSketchViewControllerDelegate defines functions which are called when the user selects one option in the AbortSketchViewController. Figure 4.16 shows the details of this delegate.

AppDelegate

This delegate is included in the iOS SDK and provides access to app specific information such as a notification when app changes state from background to foreground[21].

[19] `https://developer.apple.com/library/ios/documentation/swift/conceptual/Swift_Programming_Language/AccessControl.html`, last accessed: 20. Nov. 2014
[20] `https://developer.apple.com/library/ios/documentation/swift/conceptual/Swift_Programming_Language/Protocols.html`, last accessed: 20. Nov. 2014
[21] `https://developer.apple.com/library/ios/documentation/uikit/reference/uiapplicationdelegate_protocol/index.html`, last accessed: 09. Nov. 2014

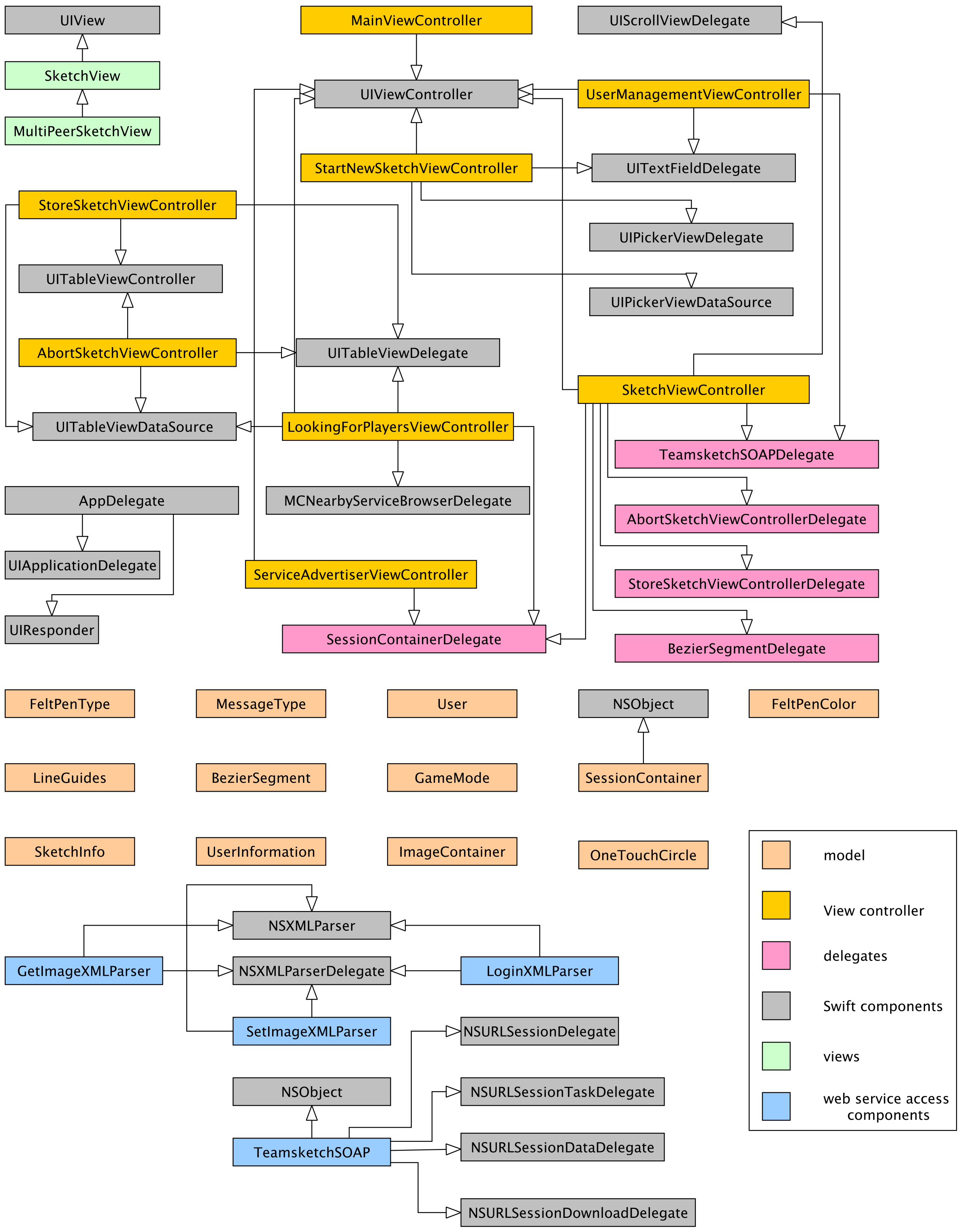


Figure 4.15: Overview of involved components

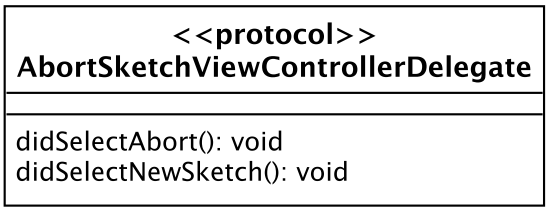


Figure 4.16: Abort Sketch View Controller Delegate

BezierSegmentDelegate

This delegate forwards already drawn bezier segments and points. Figure 4.17 shows the details of this delegate.

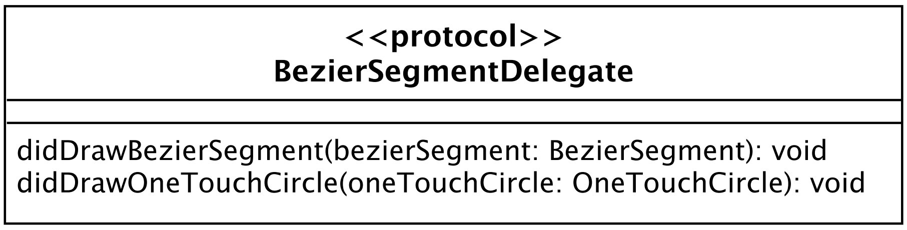

<<protocol>>
BezierSegmentDelegate
didDrawBezierSegment(bezierSegment: BezierSegment): void
didDrawOneTouchCircle(oneTouchCircle: OneTouchCircle): void

Figure 4.17: Bezier Segment Delegate

SessionContainerDelegate

This delegate forwards session related events. The corresponding delegate function will be called, when the app receives a recognized multi-peer distributed message within a session. Figure 4.18 shows the details of this delegate.

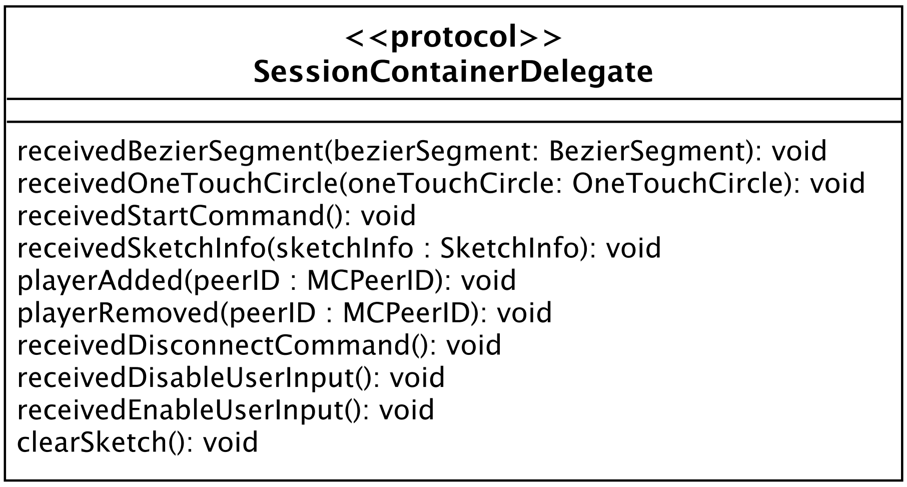

<<protocol>>
SessionContainerDelegate
receivedBezierSegment(bezierSegment: BezierSegment): void
receivedOneTouchCircle(oneTouchCircle: OneTouchCircle): void
receivedStartCommand(): void
receivedSketchInfo(sketchInfo : SketchInfo): void
playerAdded(peerID : MCPeerID): void
playerRemoved(peerID : MCPeerID): void
receivedDisconnectCommand(): void
receivedDisableUserInput(): void
receivedEnableUserInput(): void
clearSketch(): void

Figure 4.18: Session Container Delegate

StoreSketchViewControllerDelegate

The StoreSketchViewControllerDelegate defines functions which will be called when user selects one option in the StoreSketchViewController. Figure 4.19 shows the details of this delegate.

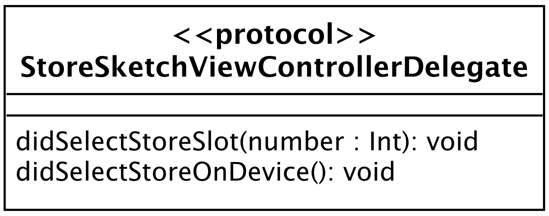
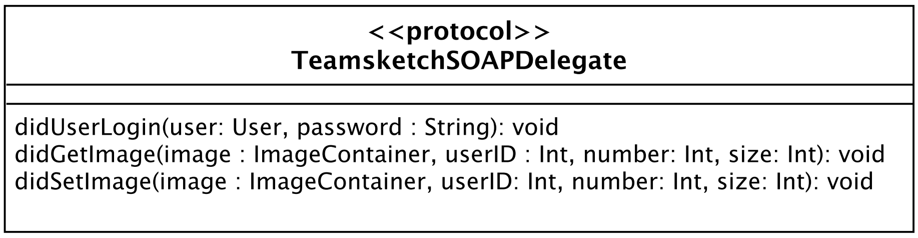

<<protocol>>
StoreSketchViewControllerDelegate
didSelectStoreSlot(number : Int): void
didSelectStoreOnDevice(): void

Figure 4.19: Store Sketch View Controller Delegate

TeamsketchSOAPDelegate

This delegate forwards the web service response. Figure 4.20 shows the details of this delegate. Every function forwards the returned SOAP[22] data of the related SOAP request.



```
<<protocol>>
TeamsketchSOAPDelegate

didUserLogin(user: User, password : String): void
didGetImage(image : ImageContainer, userID : Int, number: Int, size: Int): void
didSetImage(image : ImageContainer, userID: Int, number: Int, size: Int): void
```

Figure 4.20: Teamsketch SOAP Delegate

4.8.3 Model

All model components handle data related low level tasks.

Drawing Components

Two different shapes can be drawn in Teamsketch:

- stroke segments
- points

When the user draws a freehand stroke, the stroke starts where the user touches down the finger on the display. The stroke will be extended while the user moves the finger on the display without lifting. When the user lifts his finger, the stroke is finalized. The whole stroke is separated into stroke segments to simplify broadcasting over the multi-peer network. These segments will be used to construct cubic Bezier curves. Bezier curves are used to emulate drawn curves by adapting the location of the control polygon vertices [Farin, 2002, p. 49]. Many drawing tools, for example Gimp[23] uses this technique to create user defined paths. Bezier curves are used to optimize the stroke appearance.

[22]Simple Object Access Protocol
[23]http://www.gimp.org, last accessed: 20. Nov. 2014

Each cubic bezier curve segment has four parameters [Farin, 2002, p. 49]:

- start point
- control point 1
- control point 2
- end point

Such Bezier curves are already implemented in iOS SDK in UIBezierPath class[24]. The detailed curve fitting and smoothing algorithm will be explained in the SketchView chapter (4.8.5). Bezier segments solve the task of drawing and broadcasting strokes. When a user just taps on the sketch area, no line can be constructed. In this case the touch point will be stored as circle origin. The felt pen color as well as the felt pen width, which reflects the circle radius, is stored. The class OneTouchCircle implements this functionality. Figure 4.21 elaborates on the details of this class. The class BezierSegment holds stroke information such as stroke width, color and location to transfer the stroke information over the multi-peer network. Figure 4.22 shows the details of this class. Both classes implement a string constructor as well as an object to string converter used for transmitting the shapes over the multi-peer network.

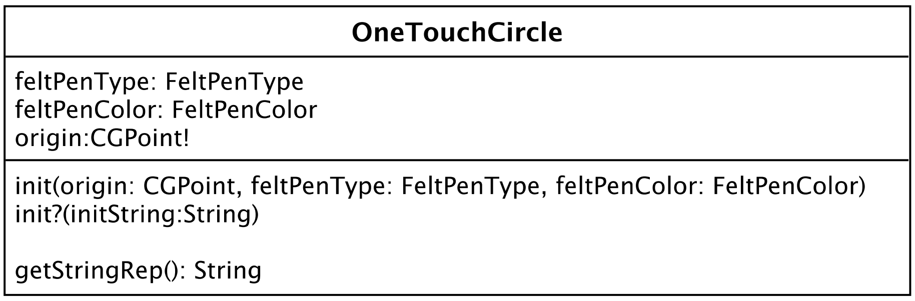
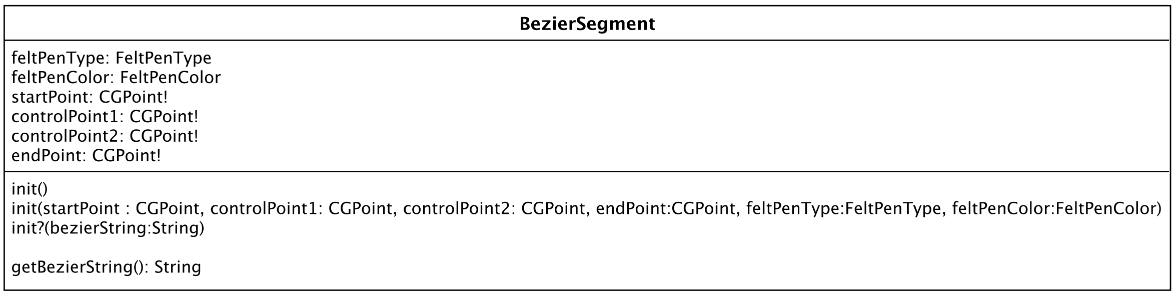

OneTouchCircle
feltPenType: FeltPenType
feltPenColor: FeltPenColor
origin:CGPoint!
init(origin: CGPoint, feltPenType: FeltPenType, feltPenColor: FeltPenColor)
init?(initString:String)
getStringRep(): String

Figure 4.21: One Touch Circle

Session Management

The session container holds the own peer ID as well as the session object and its delegate. The delegate forwards notifications when peers connect or disconnect and messages are received or sent. Teamsketch uses a session in message mode ”reliable”. The Multipeer framework takes care that all messages are transmitted to connected

[24] `https://developer.apple.com/LIBRARY/ios/documentation/UIKit/Reference/UIBezierPath_class/index.html`, last accessed: 20. Nov 2014



```
BezierSegment

feltPenType: FeltPenType
feltPenColor: FeltPenColor
startPoint: CGPoint!
controlPoint1: CGPoint!
controlPoint2: CGPoint!
endPoint: CGPoint!

init()
init(startPoint : CGPoint, controlPoint1: CGPoint, controlPoint2: CGPoint, endPoint:CGPoint, feltPenType:FeltPenType, feltPenColor:FeltPenColor)
init?(bezierString:String)

getBezierString(): String
```

Figure 4.22: Bezier segment

peers and all peers will receive the messages in correct order[25]. Figure 4.23 shows the SessionContainer details. Each message consists of a message type as Integer and a message string separated by a subscript ("_"). The following messages can be sent:

- sendBezierSegment
  This method sends a line segment (Bezier segment).

- sendOneTouchCircle
  Circles are sent if user only taps on sketch screen once (no swipe).

- sendStartCommand
  The host app sends this command after the start button was touched.

- sendSketchInfo
  This information is broadcasted to all connected peers to display game information on screen and sets the optional timer.

- sendSketchInfoToPeer
  Sketch information can also be sent to certain peers without broadcasting.

- sendDisableUserInputToSync
  This method was implemented to handle performance issues if users draw too fast and/or the network is overloaded. This disables the user input (drawing) until the drawing queue will be empty and all the necessary data was transmitted over the network.

- sendEnableUserInput
  After processing all the items in the drawing queue and the network is not overloaded any more, all users can continue their drawings. Therefore this

[25]https://developer.apple.com/library/IOs/documentation/MultipeerConnectivity/Reference/MCSessionClassRef/index.html#//apple_ref/c/tdef/MCSessionSendDataMode, last accessed: 10. Nov. 2014

command is broadcasted to all connected peers.

- sendDisconnectCommandToPeer
  If host decides to decline a peer before starting the sketch, this command is sent to the peer.

- sendClearSketchCommand
  This command is used if users decide to restart a sketch. It clears the sketch but game parameters will not be changed.

Figure 4.24 shows message examples of all types.

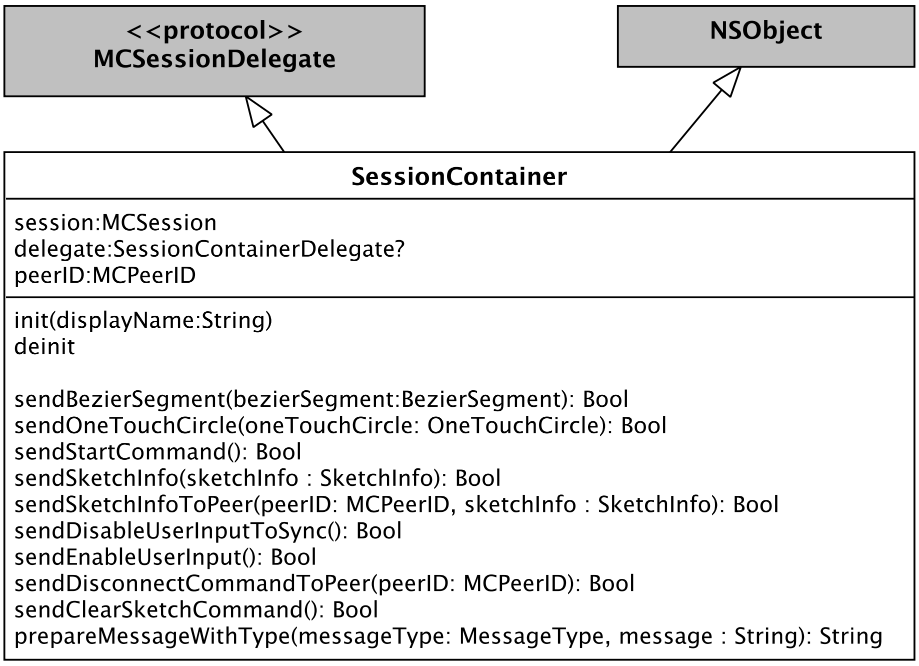

<<protocol>>
MCSessionDelegate
NSObject
SessionContainer
session:MCSession
delegate:SessionContainerDelegate?
peerID:MCPeerID
init(displayName:String)
deinit
sendBezierSegment(bezierSegment:BezierSegment): Bool
sendOneTouchCircle(oneTouchCircle: OneTouchCircle): Bool
sendStartCommand(): Bool
sendSketchInfo(sketchInfo : SketchInfo): Bool
sendSketchInfoToPeer(peerID: MCPeerID, sketchInfo : SketchInfo): Bool
sendDisableUserInputToSync(): Bool
sendEnableUserInput(): Bool
sendDisconnectCommandToPeer(peerID: MCPeerID): Bool
sendClearSketchCommand(): Bool
prepareMessageWithType(messageType: MessageType, message : String): String

Figure 4.23: Session Container

User Related Data

When the user is logged in, all user related information is stored in a UserInformation singleton. Due to security reasons the password is not stored within the singleton. Figure 4.25 shows user related component data. The user class is used to handle user information between several methods and classes.

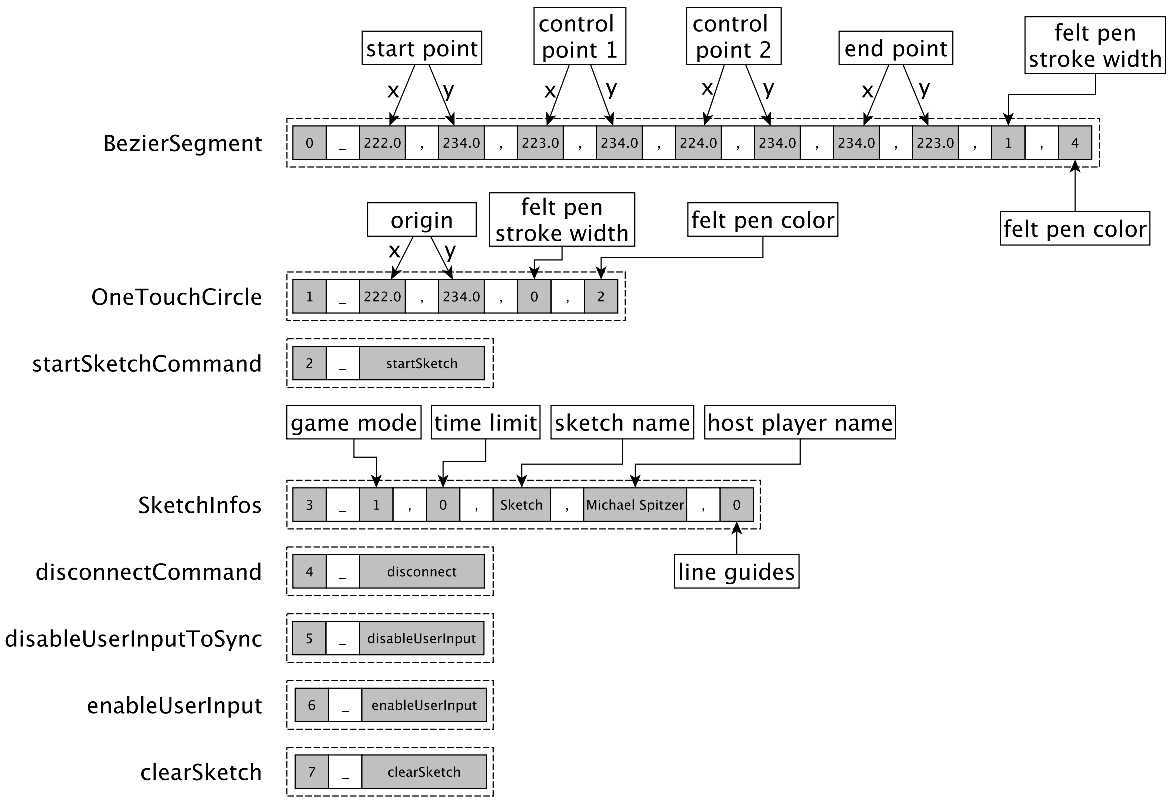


Figure 4.24: Multipeer message strings - examples

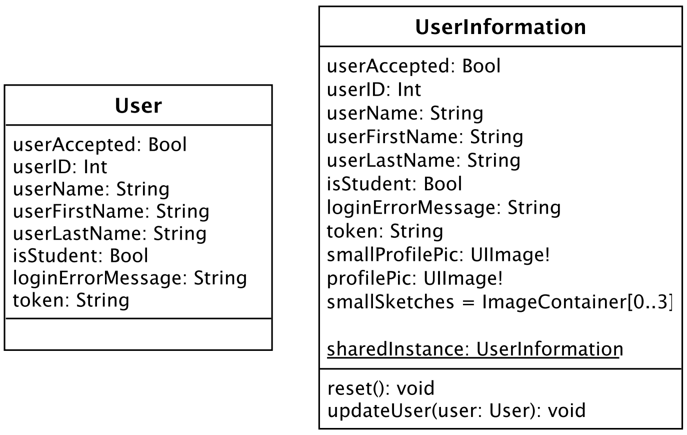


Figure 4.25: User related components

Enumerations

To simplify parameters and ensure type safety, several enumerations have been defined. Figure 4.26 shows all enumerations used in Teamsketch app. Enumeration elements are stored as integers starting with zero.

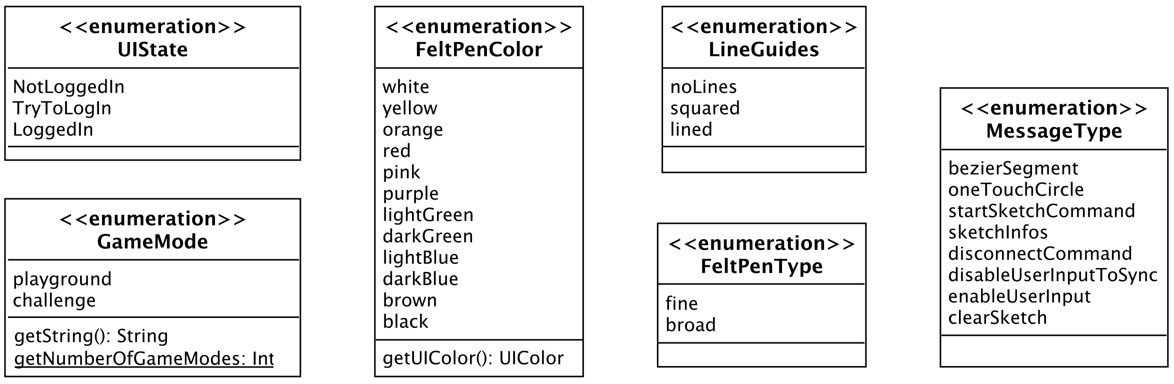


Figure 4.26: Teamsketch enumerations

4.8.4 View Controller

View Controllers implement the controller part of the MVC pattern as well as other important functionality. There are different types of view controllers. View controllers used in Teamsketch are UIViewControllers and UITableViewController which are connected with segues. These segues define transitions between different view controllers[26].

Main View Controller

This is the first view controller which is shown after the start (splash) screen. Figure 4.27 shows the details of this component.

This controller updates the user button information (user name and profile picture) if the user has already logged in and it manages segues to other view controllers.

[26]https://developer.apple.com/library/ios/featuredarticles/ViewControllerPGforiPhoneOS/AboutViewControllers/AboutViewControllers.html, last accessed: 10. Nov. 2014

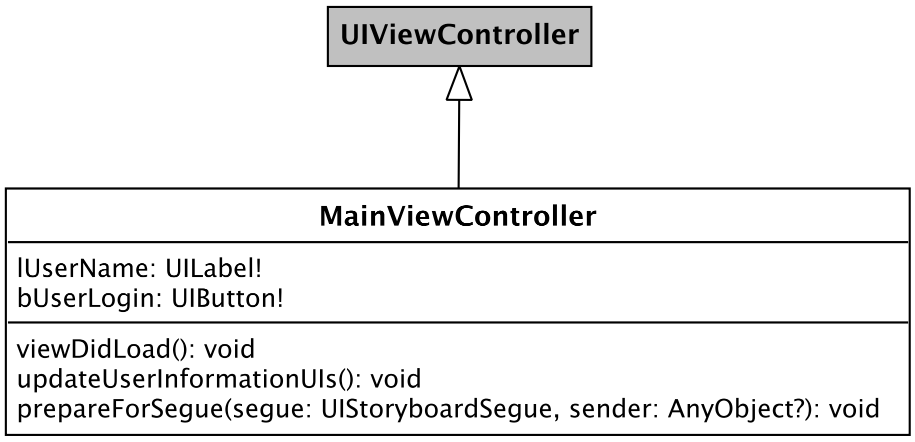

UIViewController
MainViewController
lUserName: UILabel!
bUserLogin: UIButton!
viewDidLoad(): void
updateUserInformationUIs(): void
prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?): void

Figure 4.27: Main View Controller

User Management View Controller

This controller manages user related actions such as login and logout as well as profile picture editing and storing. A TeamsketchSOAP object is used to connect the app with the web service. Figure 4.28 elaborates on the details of this component. Some properties concerning drawing (felt pens and layout constraints) have been hidden to simplify the class overview. These components will be explained in the SketchViewController chapter (4.8.4).

Start New Sketch View Controller

Various game parameters can be set within this view controller. Figure 4.29 shows the details of this view controller. After the user confirms the parameters the sketch information will be forwarded to the LookingForPlayersViewController.

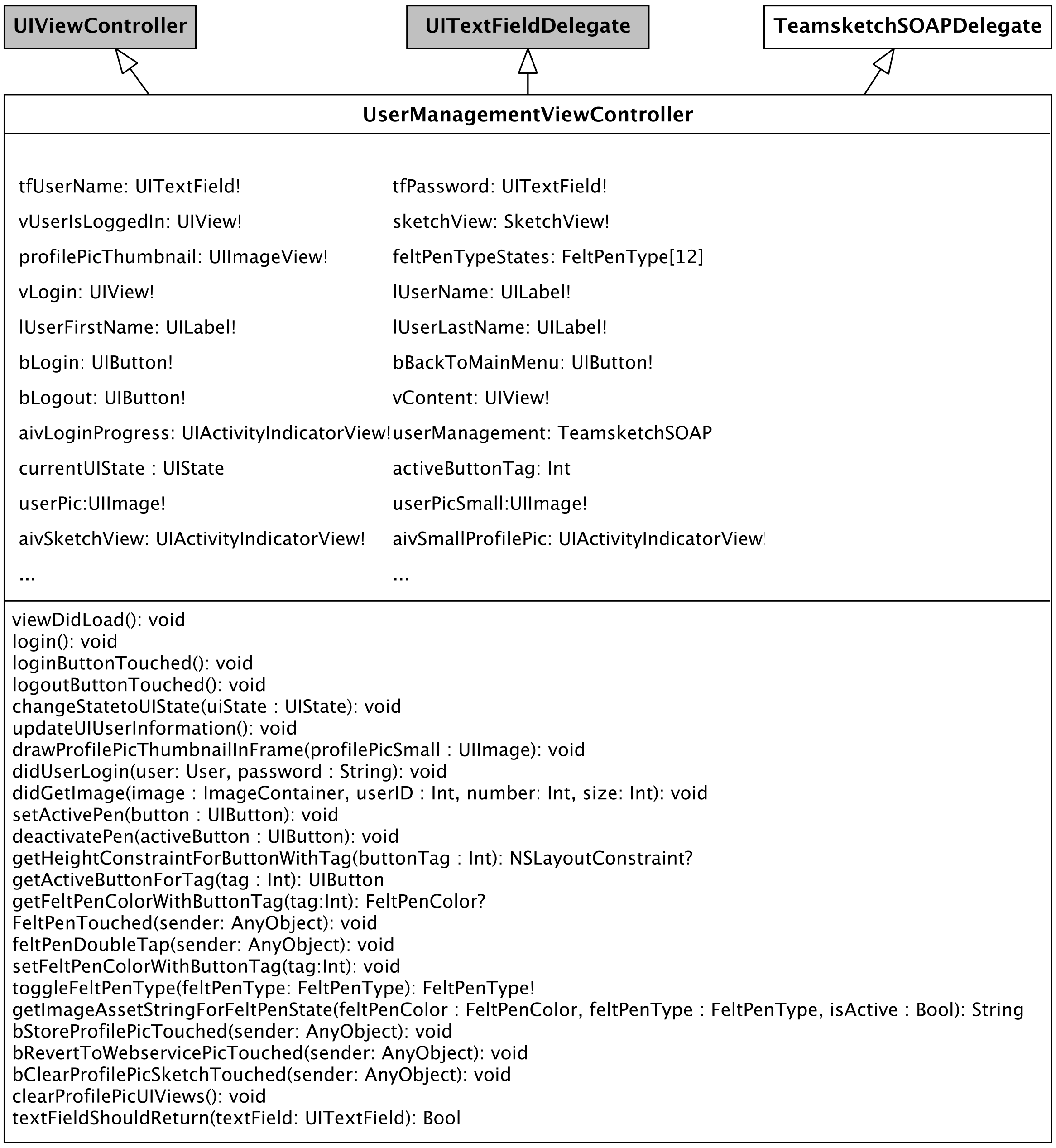

UIViewController
UITextFieldDelegate
TeamsketchSOAPDelegate
UserManagementViewController
tfUserName: UITextField!
tfPassword: UITextField!
vUserIsLoggedIn: UIView!
sketchView: SketchView!
profilePicThumbnail: UIImageView!
feltPenTypeStates: FeltPenType[12]
vLogin: UIView!
lUserName: UILabel!
lUserFirstName: UILabel!
lUserLastName: UILabel!
bLogin: UIButton!
bBackToMainMenu: UIButton!
bLogout: UIButton!
vContent: UIView!
aivLoginProgress: UIActivityIndicatorView!
userManagement: TeamsketchSOAP
currentUIState : UIState
activeButtonTag: Int
userPic:UIImage!
userPicSmall:UIImage!
aivSketchView: UIActivityIndicatorView!
aivSmallProfilePic: UIActivityIndicatorView
...
...
viewDidLoad(): void
login(): void
loginButtonTouched(): void
logoutButtonTouched(): void
changeStatetoUIState(uiState : UIState): void
updateUIUserInformation(): void
drawProfilePicThumbnailInFrame(profilePicSmall : UIImage): void
didUserLogin(user: User, password : String): void
didGetImage(image : ImageContainer, userID : Int, number: Int, size: Int): void
setActivePen(button : UIButton): void
deactivatePen(activeButton : UIButton): void
getHeightConstraintForButtonWithTag(buttonTag : Int): NSLayoutConstraint?
getActiveButtonForTag(tag : Int): UIButton
getFeltPenColorWithButtonTag(tag:Int): FeltPenColor?
FeltPenTouched(sender: AnyObject): void
feltPenDoubleTap(sender: AnyObject): void
setFeltPenColorWithButtonTag(tag:Int): void
toggleFeltPenType(feltPenType: FeltPenType): FeltPenType!
getImageAssetStringForFeltPenState(feltPenColor : FeltPenColor, feltPenType : FeltPenType, isActive : Bool): String
bStoreProfilePicTouched(sender: AnyObject): void
bRevertToWebservicePicTouched(sender: AnyObject): void
bClearProfilePicSketchTouched(sender: AnyObject): void
clearProfilePicUIViews(): void
textFieldShouldReturn(textField: UITextField): Bool

Figure 4.28: User Management View Controller

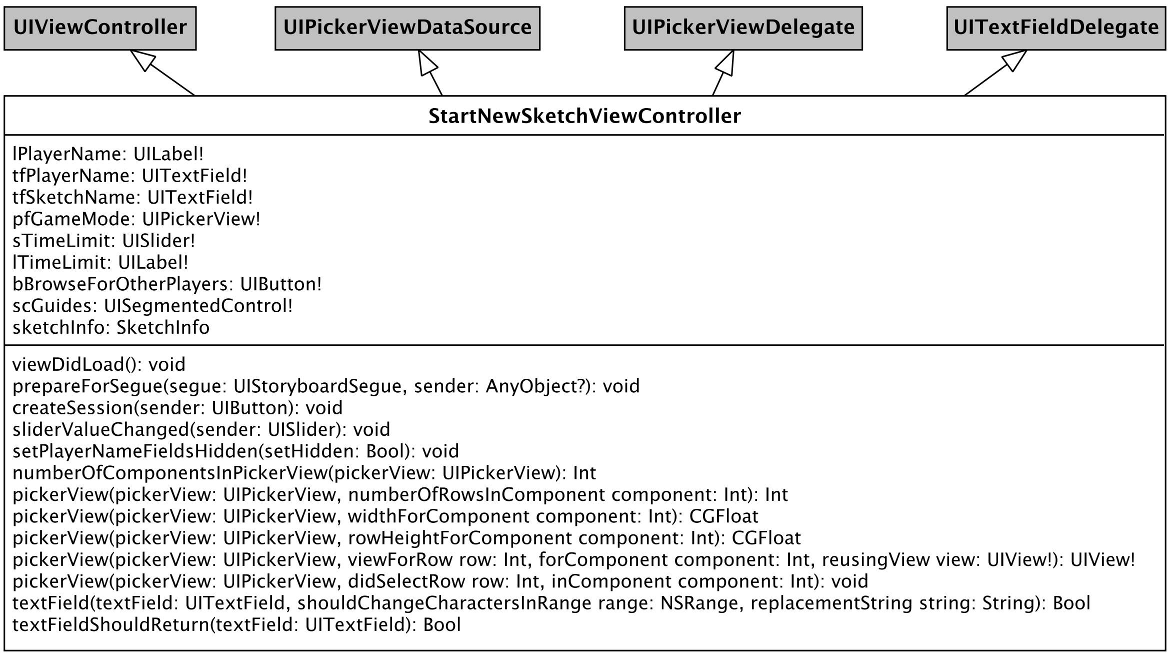

UIViewController
UIPickerViewDataSource
UIPickerViewDelegate
UITextFieldDelegate
StartNewSketchViewController
lPlayerName: UILabel!
tfPlayerName: UITextField!
tfSketchName: UITextField!
pfGameMode: UIPickerView!
sTimeLimit: UISlider!
lTimeLimit: UILabel!
bBrowseForOtherPlayers: UIButton!
scGuides: UISegmentedControl!
sketchInfo: SketchInfo
viewDidLoad(): void
prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?): void
createSession(sender: UIButton): void
sliderValueChanged(sender: UISlider): void
setPlayerNameFieldsHidden(setHidden: Bool): void
numberOfComponentsInPickerView(pickerView: UIPickerView): Int
pickerView(pickerView: UIPickerView, numberOfRowsInComponent component: Int): Int
pickerView(pickerView: UIPickerView, widthForComponent component: Int): CGFloat
pickerView(pickerView: UIPickerView, rowHeightForComponent component: Int): CGFloat
pickerView(pickerView: UIPickerView, viewForRow row: Int, forComponent component: Int, reusingView view: UIView!): UIView!
pickerView(pickerView: UIPickerView, didSelectRow row: Int, inComponent component: Int): void
textField(textField: UITextField, shouldChangeCharactersInRange range: NSRange, replacementString string: String): Bool
textFieldShouldReturn(textField: UITextField): Bool

Figure 4.29: Start New Sketch View Controller

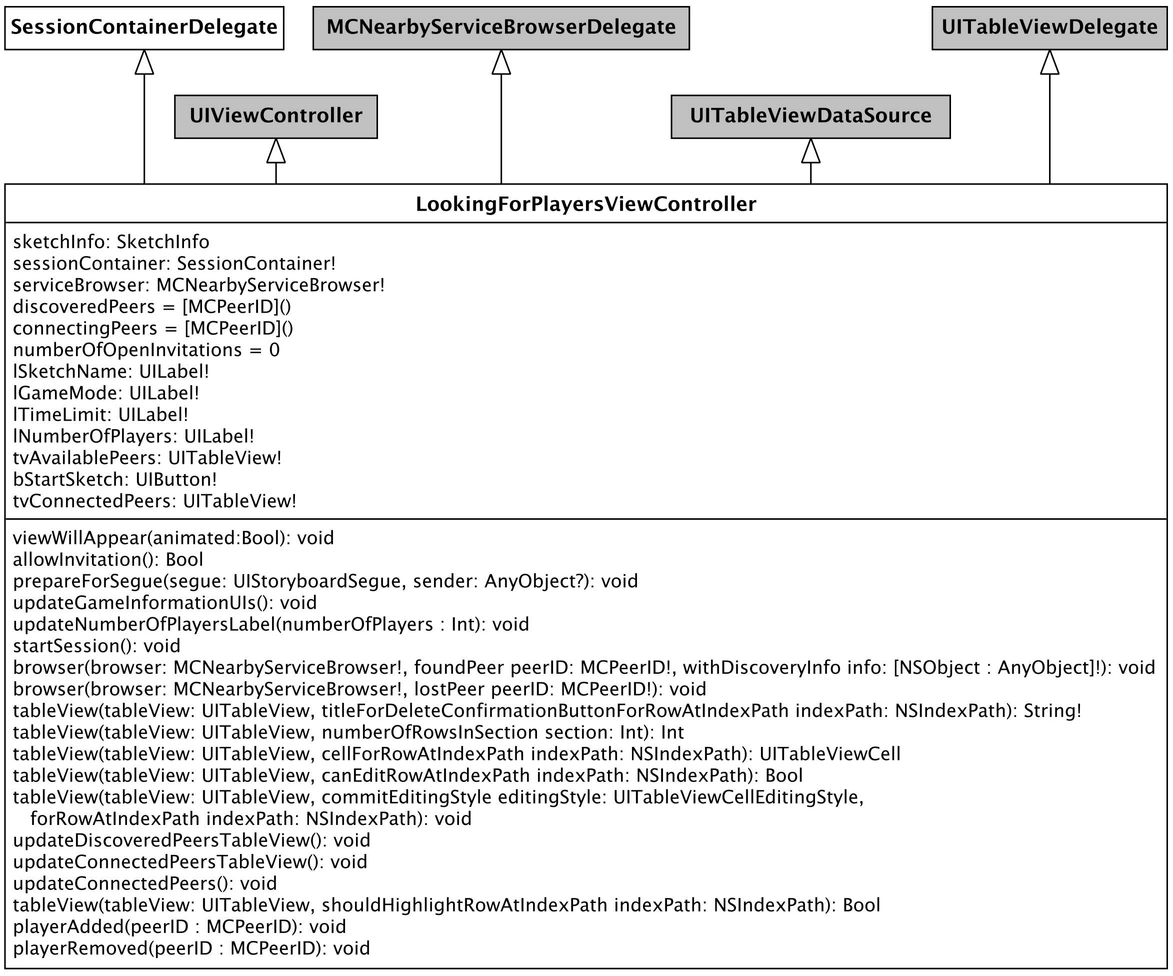
Looking For Players View Controller

This view controller creates the multi-peer session and manages the peers. Information about connected and disconnected peers are forwarded by the session container delegate. The controller manages two tables. The first table lists the peers in range. This list is filled by the MCNearbyServiceBrowserDelegate which forwards found peers and lost peers to keep the peer list up to date. The SessionContainerDelegate is used to keep the second list (already connected peers) up to date. This update process is done as follows:

- peer connected:
  - send sketch info to new connected peer
  - update number of connected peers
  - update table views

- peer disconnected
  - update number of connected peers
  - update table views

Figure 4.30 elaborates on the details of this controller.



```
sketchInfo: SketchInfo
sessionContainer: SessionContainer!
serviceBrowser: MCNearbyServiceBrowser!
discoveredPeers = [MCPeerID]()
connectingPeers = [MCPeerID]()
numberOfOpenInvitations = 0
lSketchName: UILabel!
lGameMode: UILabel!
lTimeLimit: UILabel!
lNumberOfPlayers: UILabel!
tvAvailablePeers: UITableView!
bStartSketch: UIButton!
tvConnectedPeers: UITableView!
```

```
viewWillAppear(animated:Bool): void
allowInvitation(): Bool
prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?): void
updateGameInformationUIs(): void
updateNumberOfPlayersLabel(numberOfPlayers : Int): void
startSession(): void
browser(browser: MCNearbyServiceBrowser!, foundPeer peerID: MCPeerID!, withDiscoveryInfo info: [NSObject : AnyObject]!): void
browser(browser: MCNearbyServiceBrowser!, lostPeer peerID: MCPeerID!): void
tableView(tableView: UITableView, titleForDeleteConfirmationButtonForRowAtIndexPath indexPath: NSIndexPath): String!
tableView(tableView: UITableView, numberOfRowsInSection section: Int): Int
tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath): UITableViewCell
tableView(tableView: UITableView, canEditRowAtIndexPath indexPath: NSIndexPath): Bool
tableView(tableView: UITableView, commitEditingStyle editingStyle: UITableViewCellEditingStyle,
  forRowAtIndexPath indexPath: NSIndexPath): void
updateDiscoveredPeersTableView(): void
updateConnectedPeersTableView(): void
updateConnectedPeers(): void
tableView(tableView: UITableView, shouldHighlightRowAtIndexPath indexPath: NSIndexPath): Bool
playerAdded(peerID : MCPeerID): void
playerRemoved(peerID : MCPeerID): void
```

Figure 4.30: Looking For Players View Controller

Service Advertiser View Controller

This view controller manages multi-peer advertising. The SessionContainerDelegate forwards broadcasted host commands. Figure 4.31 elaborates on the details of this controller.

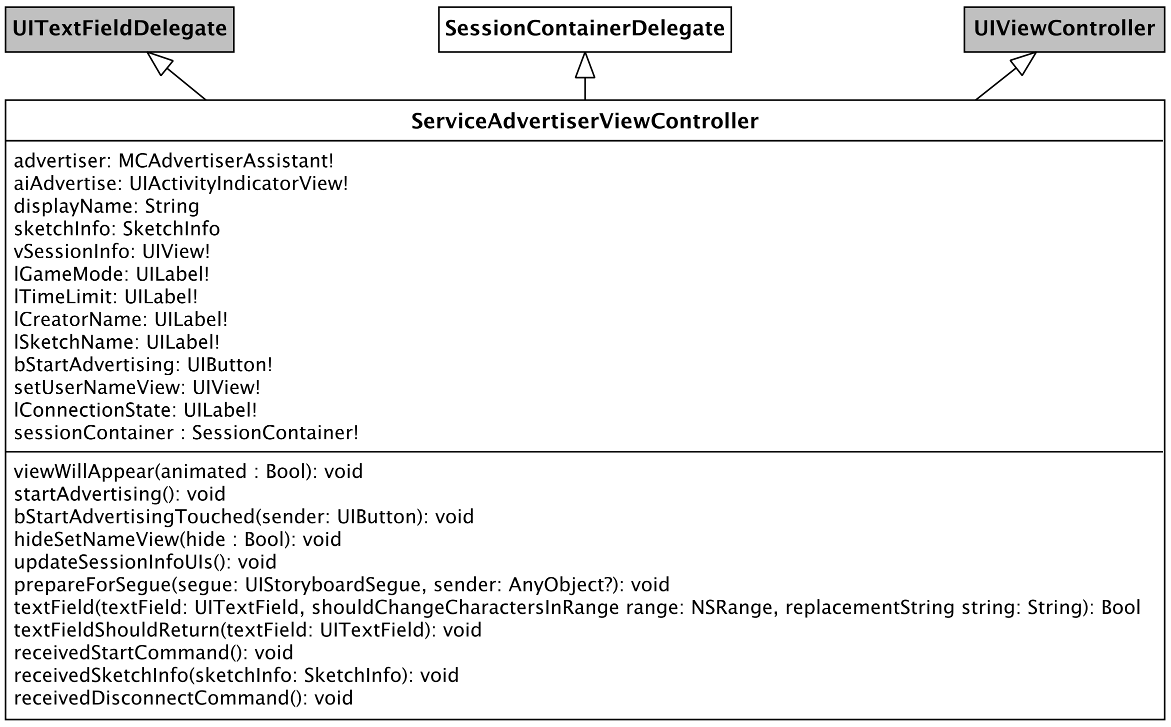
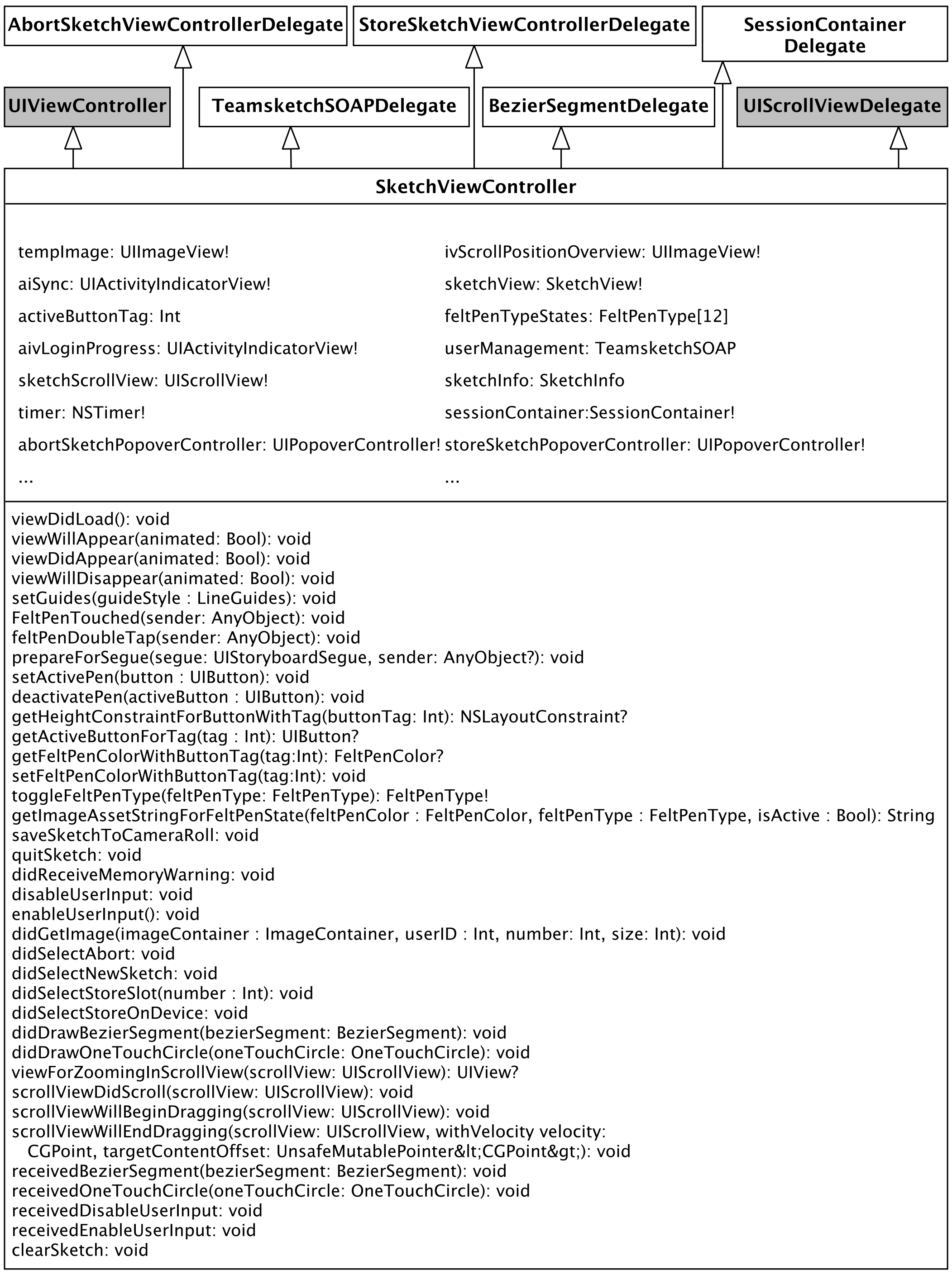

UITextFieldDelegate
SessionContainerDelegate
UIViewController
ServiceAdvertiserViewController
advertiser: MCAdvertiserAssistant!
aiAdvertise: UIActivityIndicatorView!
displayName: String
sketchInfo: SketchInfo
vSessionInfo: UIView!
lGameMode: UILabel!
lTimeLimit: UILabel!
lCreatorName: UILabel!
lSketchName: UILabel!
bStartAdvertising: UIButton!
setUserNameView: UIView!
lConnectionState: UILabel!
sessionContainer : SessionContainer!
viewWillAppear(animated : Bool): void
startAdvertising(): void
bStartAdvertisingTouched(sender: UIButton): void
hideSetNameView(hide : Bool): void
updateSessionInfoUIs(): void
prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?): void
textField(textField: UITextField, shouldChangeCharactersInRange range: NSRange, replacementString string: String): Bool
textFieldShouldReturn(textField: UITextField): void
receivedStartCommand(): void
receivedSketchInfo(sketchInfo: SketchInfo): void
receivedDisconnectCommand(): void

Figure 4.31: Service Advertiser View Controller

Sketch View Controller

This view controller manages all sketch related tasks. When the view appears and the user is logged in, user management tries to fetch stored sketch thumbnails from the web service. These thumbnails are used in the StoreSketchViewController Pop-up. If user selects an option in one of the Pop-up Controller (StoreSketchViewController or AbortSketchViewController), the corresponding delegate forwards the selected option to the SketchViewController. The SessionContainerDelegate forwards received sketch data and commands to the Sketch View. The sketch view is embedded in a UIScrollView to enable scrolling and zooming. When the sketch view drew a Bezier segment or a one touch circle, the shapes are forwarded to the Session Container send queue by the Bezier Segment Delegate.
The SketchViewController implements logic for changing felt pens in the felt pen color bar. Select felt pen animations are implemented as height constraint changes.

Information about active felt pen width and color as well as line guide information is forwarded to the Sketch View. Figure 4.32 shows the details of this view controller. The felt pen buttons and height constraints are removed to simplify the figure.



```
tempImage: UIImageView!                              ivScrollPositionOverview: UIImageView!
aiSync: UIActivityIndicatorView!                     sketchView: SketchView!
activeButtonTag: Int                                 feltPenTypeStates: FeltPenType[12]
aivLoginProgress: UIActivityIndicatorView!           userManagement: TeamsketchSOAP
sketchScrollView: UIScrollView!                      sketchInfo: SketchInfo
timer: NSTimer!                                      sessionContainer:SessionContainer!
abortSketchPopoverController: UIPopoverController!   storeSketchPopoverController: UIPopoverController!
...                                                  ...
```

```
viewDidLoad(): void
viewWillAppear(animated: Bool): void
viewDidAppear(animated: Bool): void
viewWillDisappear(animated: Bool): void
setGuides(guideStyle : LineGuides): void
FeltPenTouched(sender: AnyObject): void
feltPenDoubleTap(sender: AnyObject): void
prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?): void
setActivePen(button : UIButton): void
deactivatePen(activeButton : UIButton): void
getHeightConstraintForButtonWithTag(buttonTag: Int): NSLayoutConstraint?
getActiveButtonForTag(tag : Int): UIButton?
getFeltPenColorWithButtonTag(tag:Int): FeltPenColor?
setFeltPenColorWithButtonTag(tag:Int): void
toggleFeltPenType(feltPenType: FeltPenType): FeltPenType!
getImageAssetStringForFeltPenState(feltPenColor : FeltPenColor, feltPenType : FeltPenType, isActive : Bool): String
saveSketchToCameraRoll: void
quitSketch: void
didReceiveMemoryWarning: void
disableUserInput: void
enableUserInput(): void
didGetImage(imageContainer : ImageContainer, userID : Int, number: Int, size: Int): void
didSelectAbort: void
didSelectNewSketch: void
didSelectStoreSlot(number : Int): void
didSelectStoreOnDevice: void
didDrawBezierSegment(bezierSegment: BezierSegment): void
didDrawOneTouchCircle(oneTouchCircle: OneTouchCircle): void
viewForZoomingInScrollView(scrollView: UIScrollView): UIView?
scrollViewDidScroll(scrollView: UIScrollView): void
scrollViewWillBeginDragging(scrollView: UIScrollView): void
scrollViewWillEndDragging(scrollView: UIScrollView, withVelocity velocity:
  CGPoint, targetContentOffset: UnsafeMutablePointer&lt;CGPoint&gt;): void
receivedBezierSegment(bezierSegment: BezierSegment): void
receivedOneTouchCircle(oneTouchCircle: OneTouchCircle): void
receivedDisableUserInput: void
receivedEnableUserInput: void
clearSketch: void
```

Figure 4.32: Sketch View Controller

Pop-up Controller

AbortSketchViewController and StoreSketchViewController are implemented as UITableViewController which delegates user selections to the Sketch View Controller. Figure 4.33 elaborates on the details of these two controllers.

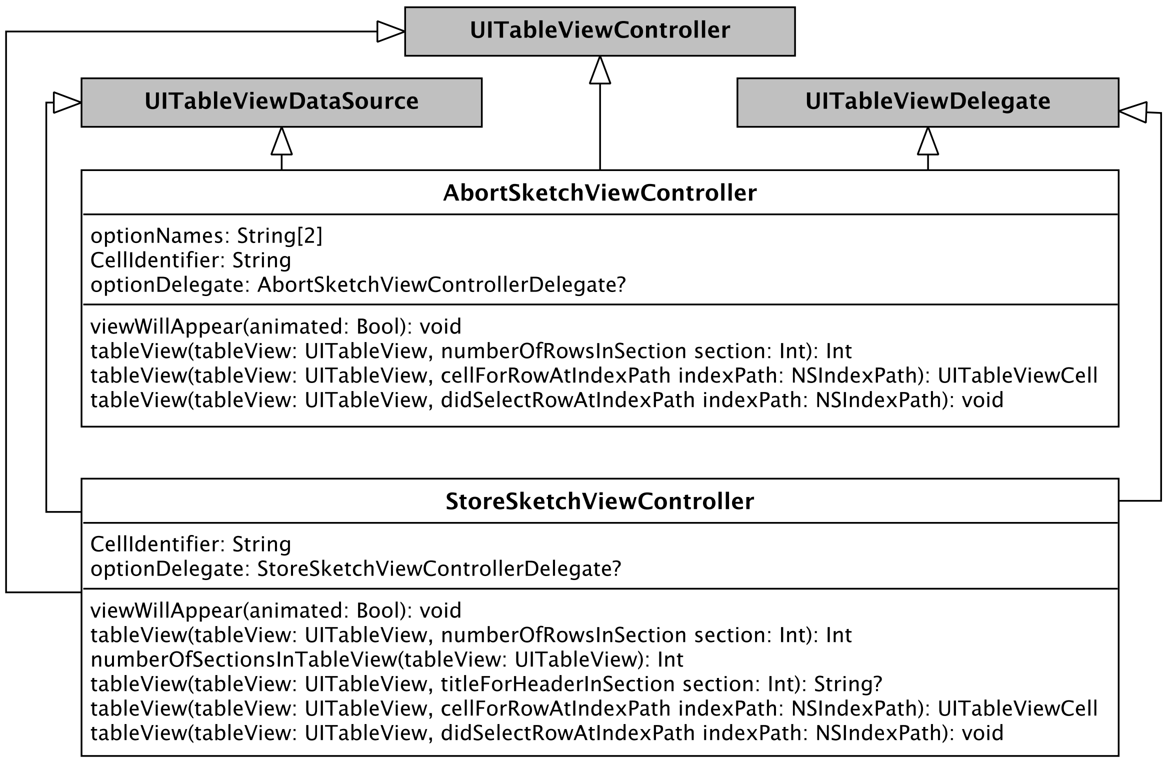

UITableViewController
UITableViewDataSource
UITableViewDelegate
AbortSketchViewController
optionNames: String[2]
CellIdentifier: String
optionDelegate: AbortSketchViewControllerDelegate?
viewWillAppear(animated: Bool): void
tableView(tableView: UITableView, numberOfRowsInSection section: Int): Int
tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath): UITableViewCell
tableView(tableView: UITableView, didSelectRowAtIndexPath indexPath: NSIndexPath): void
StoreSketchViewController
CellIdentifier: String
optionDelegate: StoreSketchViewControllerDelegate?
viewWillAppear(animated: Bool): void
tableView(tableView: UITableView, numberOfRowsInSection section: Int): Int
numberOfSectionsInTableView(tableView: UITableView): Int
tableView(tableView: UITableView, titleForHeaderInSection section: Int): String?
tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath): UITableViewCell
tableView(tableView: UITableView, didSelectRowAtIndexPath indexPath: NSIndexPath): void

Figure 4.33: Abort Sketch- and Store Sketch View Controller

4.8.5 Views

Two customized UIViews were implemented to provide sketch functionality. SketchView implements basic drawing functionality. MultiPeerSketchView adds multi-peer functionality. Figure 4.34 shows details of these two views.

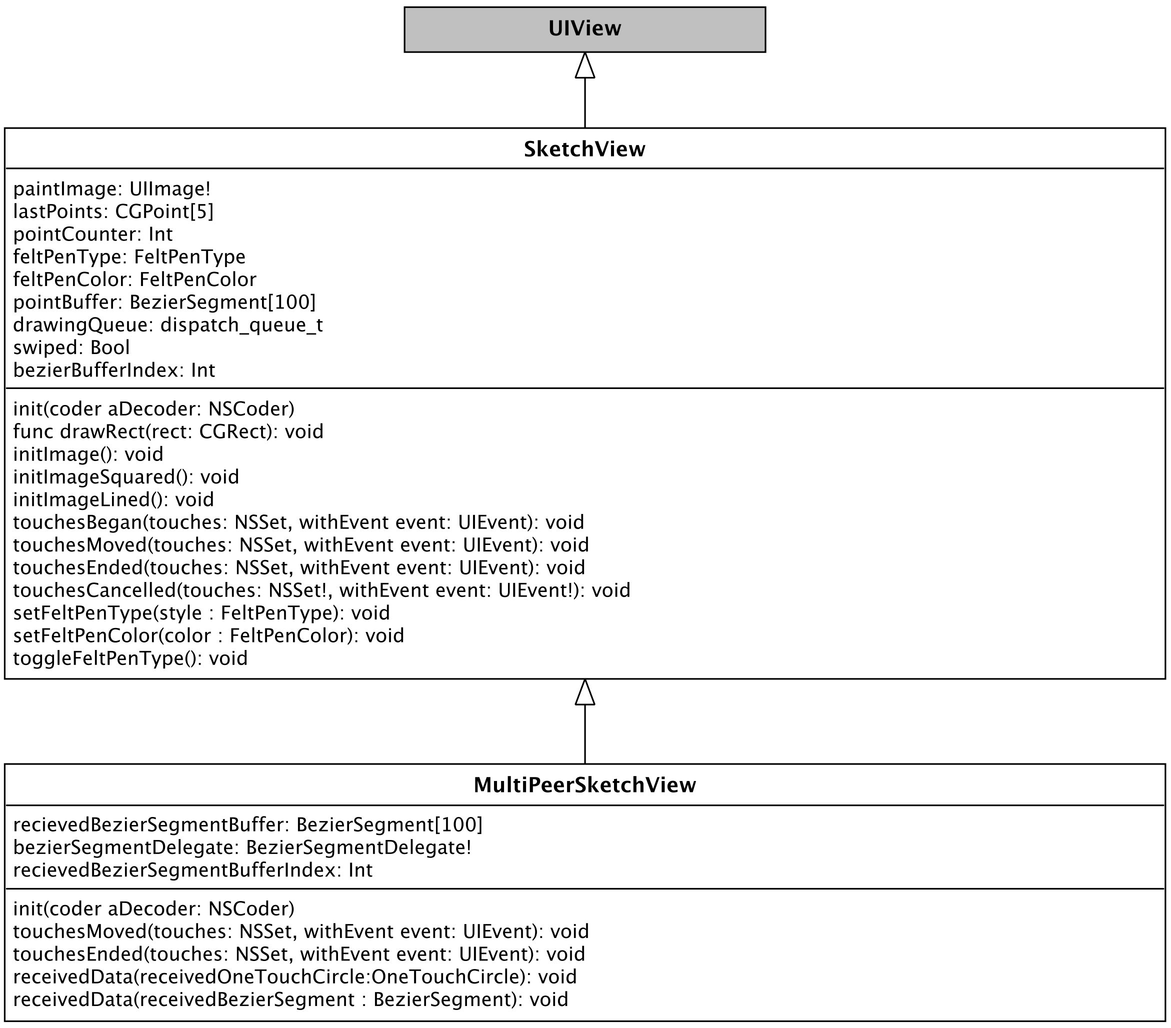

UIView
SketchView
paintImage: UIImage!
lastPoints: CGPoint[5]
pointCounter: Int
feltPenType: FeltPenType
feltPenColor: FeltPenColor
pointBuffer: BezierSegment[100]
drawingQueue: dispatch_queue_t
swiped: Bool
bezierBufferIndex: Int
init(coder aDecoder: NSCoder)
func drawRect(rect: CGRect): void
initImage(): void
initImageSquared(): void
initImageLined(): void
touchesBegan(touches: NSSet, withEvent event: UIEvent): void
touchesMoved(touches: NSSet, withEvent event: UIEvent): void
touchesEnded(touches: NSSet, withEvent event: UIEvent): void
touchesCancelled(touches: NSSet!, withEvent event: UIEvent!): void
setFeltPenType(style : FeltPenType): void
setFeltPenColor(color : FeltPenColor): void
toggleFeltPenType(): void
MultiPeerSketchView
recievedBezierSegmentBuffer: BezierSegment[100]
bezierSegmentDelegate: BezierSegmentDelegate!
recievedBezierSegmentBufferIndex: Int
init(coder aDecoder: NSCoder)
touchesMoved(touches: NSSet, withEvent event: UIEvent): void
touchesEnded(touches: NSSet, withEvent event: UIEvent): void
receivedData(receivedOneTouchCircle:OneTouchCircle): void
receivedData(receivedBezierSegment : BezierSegment): void

Figure 4.34: Sketch Views

SketchView

The SketchView holds a UImage as storage. The drawRect method was overridden to perform drawing updates when setNeedsDisplay is called. Four UIView touch methods were overridden to provide touch functionality.

- touchesBegan(...)
  This method is called when user touches a location in sketch view.

- touchesMoved(...)
  This method is continually called while user swipes on the sketch view.

- touchesEnded(...)
  When user lifts his finger, this method is called.

- touchesCancelled(...)
  This method is forwarded to touchesEnded(...).

A user swipe (touchesBegan - touchesMoved - touchesEnded) is considered as one stroke. This stroke is separated into stroke segments. The basic approach is to just store the swiped coordinates. Figure 4.35 shows a drawing example with a broad felt pen without optimization.

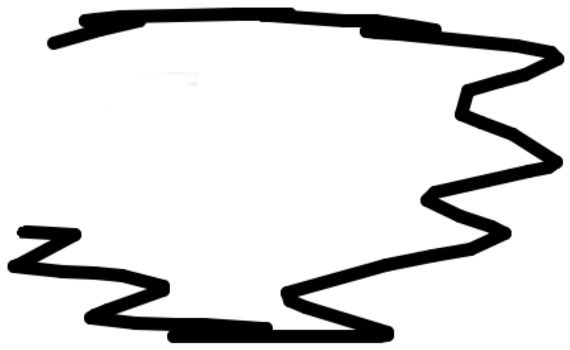


Figure 4.35: Basic drawing approach

The resulting stroke can be very angular shaped, depending on the user's swipe. A drawn stroke should not be cornered therefore the basic approach is not suitable for a drawing application. The next approach was to convert a user swipe to a Bezier curve. Figure 4.36 shows the sampled user touch points (big black dots) and the computed cubic bezier curve.

These curves had been used for the improved version. Every four points, a bezier curve was constructed and drawn. The stroke appearance improved significantly but one issue remained: when the next sampled point after the endpoint of a bezier segment is

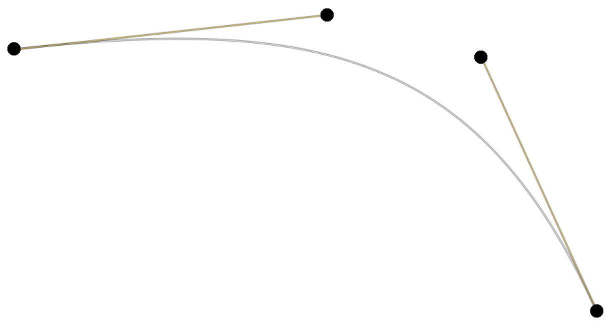


Figure 4.36: Cubic bezier curve computed with four sampled touch points

not collinear to the last two points, the stroke gets a point of discontinuity. Figure 4.37 elaborates on the details of this issue.

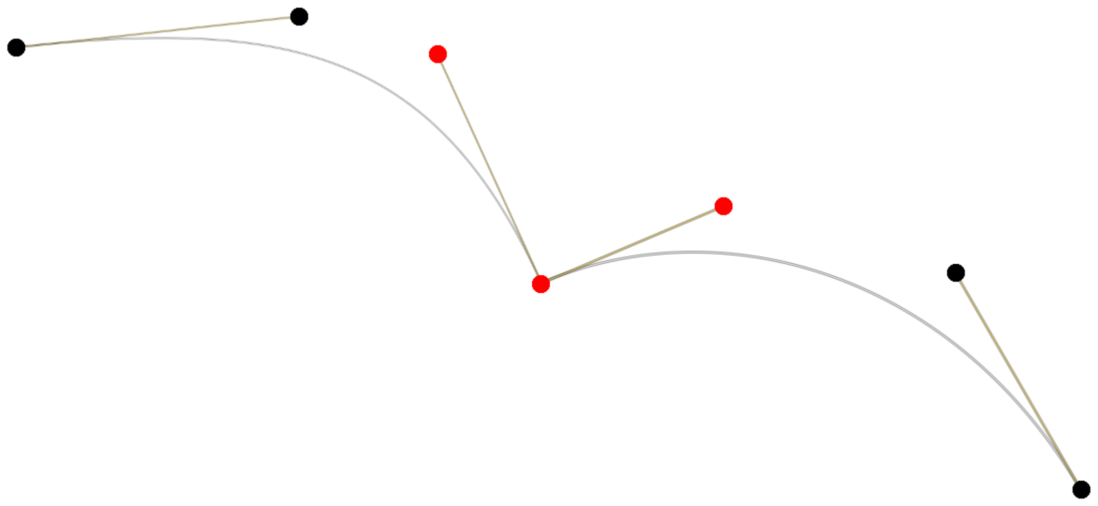


Figure 4.37: Cubic bezier curve segments with a point of discontinuity

The last point of a Bezier segment which is the first point of the next Bezier segment at once, is critical. One approach to address this issue is to insert two additional points and avoid points of discontinuity by applying de Casteljau algorithm to the critical point and its new inserted neighbors [Hongping and Zhaoyu, 2012, p. 283]. The touch sampling rate in UIViews is very high therefore the sampled points are very close to each other. In this case linear interpolation is enough to get a good stroke appearance (tested with iPad2 and iPad3 (high resolution)). To prevent a point of discontinuity, the critical point has to be collinear to its direct neighbors. This could be reached by mapping the point on the direct line between the neighbors (linear interpolation) [Farin, 2002, p. 26]. In Bezier context it is sufficient that the observed points are collinear therefore computation could be simplified to equation 4.1. Visual observations of the stroke shape showed that this simplification does not harm the stroke appearance.

$$p_{collinear} = \frac{p_{predecessor} + p_{successor}}{2} \tag{4.1}$$

Figure 4.38 shows the result of the linear interpolation (green dot) and the drawn path.

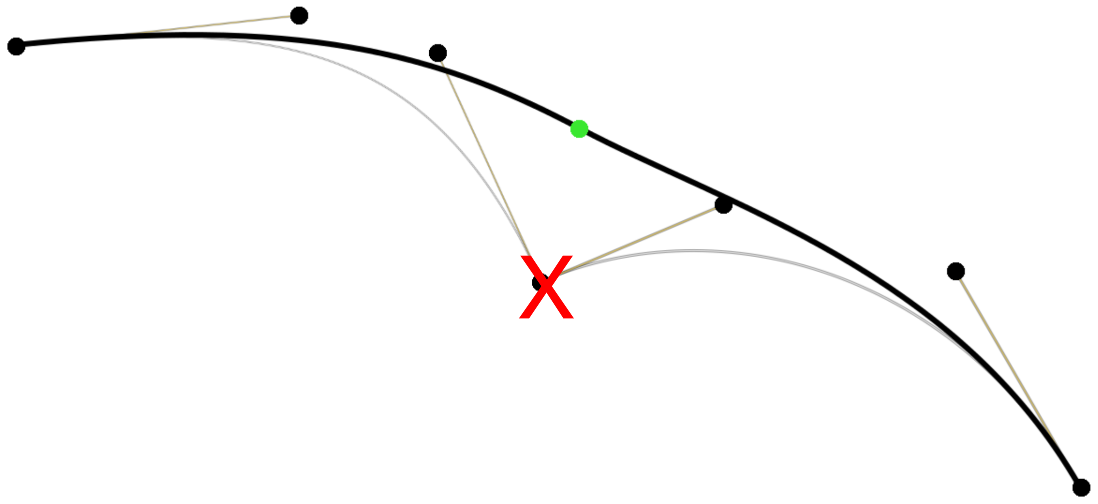


Figure 4.38: Cubic bezier curve segments with linear interpolation

Figure 4.39 shows a drawing example after the stroke curve smoothing algorithm was implemented. The stroke's appearance improved significantly compared to the basic draw approach.

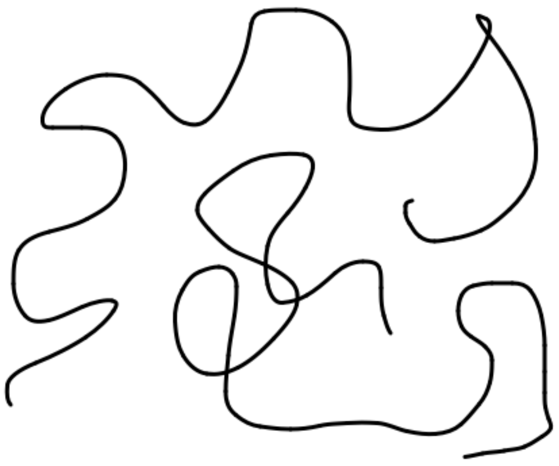
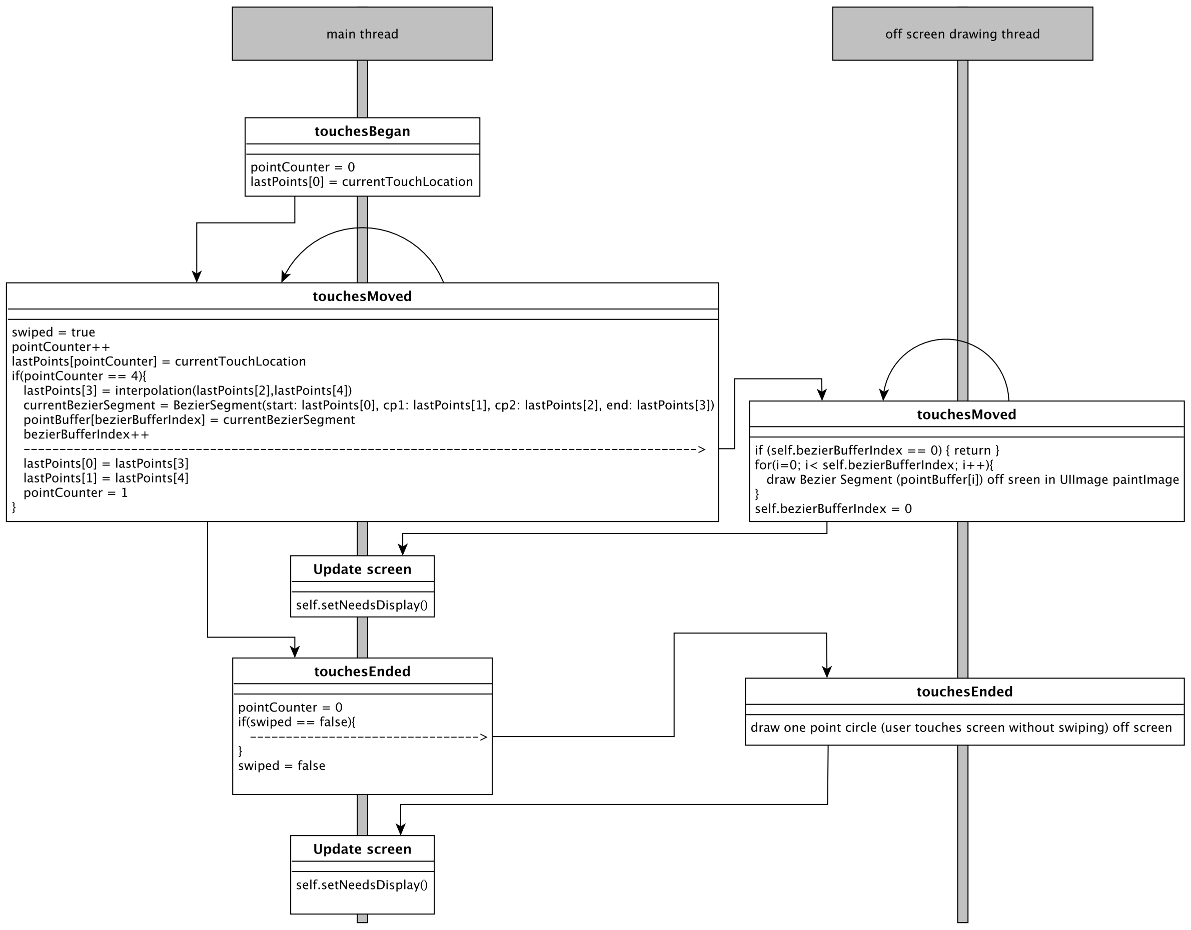


Figure 4.39: Optimized drawing path

After line (Bezier curve) optimization, the curve segment is stored in a BezierSegment for distributing over the multi-peer network. These computations are very challenging for the iPad. Therefore they can not be done on the main thread which fills the point queue as well. When 5 points are sampled, the fourth point is shifted with linear interpolation. Then the first four points are stored into a Bezier segment and sent to the Bezier segment buffer. The fourth and fifth sampled points are used as first and second point of the next Bezier segment. Immediately after a new Bezier segment has been added to the segment drawing buffer, a drawing block is added to the second thread, which does the off screen drawing. After all drawing is performed, the drawing block switches to the main thread to update the view with the modified sketch image (UIImage). Figure 4.40 shows the computation details in pseudo code.



```
touchesBegan
pointCounter = 0
lastPoints[0] = currentTouchLocation
```

```
touchesMoved
swiped = true
pointCounter++
lastPoints[pointCounter] = currentTouchLocation
if(pointCounter == 4){
   lastPoints[3] = interpolation(lastPoints[2],lastPoints[4])
   currentBezierSegment = BezierSegment(start: lastPoints[0], cp1: lastPoints[1], cp2: lastPoints[2], end: lastPoints[3])
   pointBuffer[bezierBufferIndex] = currentBezierSegment
   bezierBufferIndex++
   -------------------------------------------------------------------------------------------->
   lastPoints[0] = lastPoints[3]
   lastPoints[1] = lastPoints[4]
   pointCounter = 1
}
```

```
touchesMoved
if (self.bezierBufferIndex == 0) { return }
for(i=0; i< self.bezierBufferIndex; i++){
   draw Bezier Segment (pointBuffer[i]) off sreen in UIImage paintImage
}
self.bezierBufferIndex = 0
```

```
Update screen
self.setNeedsDisplay()
```

```
touchesEnded
pointCounter = 0
if(swiped == false){
   -------------------------------->
}
swiped = false
```

```
touchesEnded
draw one point circle (user touches screen without swiping) off screen
```

```
Update screen
self.setNeedsDisplay()
```

Figure 4.40: Drawing thread overview

MultipeerSketchView

This view inherits functionality from SketchView. When a Bezier segment was created, it will be forwarded to the BezierSegmentDelegate, which forwards the segment to the multi-peer session send queue.

4.8.6 Web service access

Teamsketch uses a SOAP web service to retrieve and store sketches online. This service will be explained in detail in chapter 5. The TeamsketchSOAP class implements all necessary functionality to communicate with the web service. XML requests and responses are used as data transmission format. XML is a worldwide established standard which used as standard transmission file format for SOAP web services [MacIntyre, Danchilla, and Gogala, 2011, p. 323]. All requests are preformed asynchronous. The response is forwarded to the TeamsketchSOAPDelegate. Sketches are transmitted as Base64 encoded String. Image encoding and decoding will be explained in the web service chapter 5 in detail. Figure 4.41 elaborates on the details of this class.

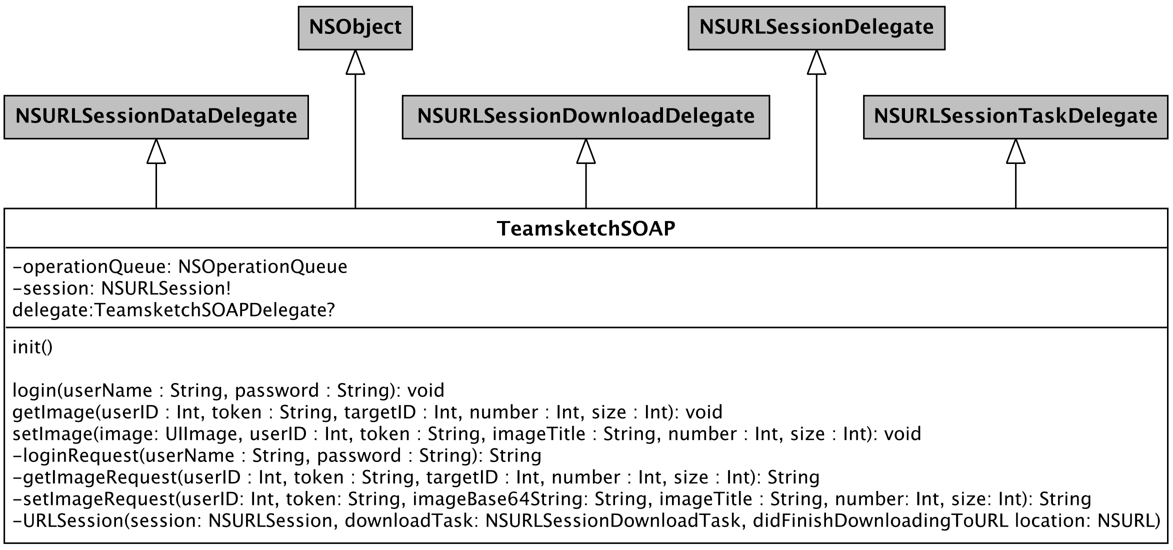

NSObject
NSURLSessionDelegate
NSURLSessionDataDelegate
NSURLSessionDownloadDelegate
NSURLSessionTaskDelegate
TeamsketchSOAP
-operationQueue: NSOperationQueue
-session: NSURLSession!
delegate:TeamsketchSOAPDelegate?
init()
login(userName : String, password : String): void
getImage(userID : Int, token : String, targetID : Int, number : Int, size : Int): void
setImage(image: UIImage, userID : Int, token : String, imageTitle : String, number : Int, size : Int): void
-loginRequest(userName : String, password : String): String
-getImageRequest(userID : Int, token : String, targetID : Int, number : Int, size : Int): String
-setImageRequest(userID: Int, token: String, imageBase64String: String, imageTitle : String, number: Int, size: Int): String
-URLSession(session: NSURLSession, downloadTask: NSURLSessionDownloadTask, didFinishDownloadingToURL location: NSURL)

Figure 4.41: TeamsketchSOAP overview

4.9 Unit Testing

The used IDE xCode 6 supports unit testing by subclassing XCTest. Apple's test automation framework supports defining expected results with assertion API and separating tests into several test methods. These methods can be grouped into test suites which can be started as one single operation. With these features the test suite belongs to the xUnit family [Meszaros, 2007, p. 76].
When creating a new App a test target will be added automatically. XCTest supports asynchronous testing which can be used for delegate callbacks or network requests. This functionality is implemented by expectation objects, which describe defined events. XCTestCase waits for this expectation to become true or for receiving a time out [Callahan and Turner, 2014]. The next section explains the used test patterns.

4.9.1 Mocks

In some cases a real code fragment (object) is substituted by a mock to observe and test indirect outputs [Meszaros, 2007, p. 137]. This principle is elaborated in detail in the following test case example. Testing the message construction can not be realized straight forward, because the session object creates the message and sends it immediately. This work flow should be tested accurately. For this reason the "sendData(...)" method in MCSession class was overridden to not send the constructed message but stores the last message data as a property. Figure 4.42 shows the details of the original classes compared to the mock classes. Only changed properties and methods are listed in the mock classes. The session object is a member of the SessionContainer class therefore the SessionContainer had to be changed to a mock as well. Listing 4.3 shows a SessionContainer Mock test example.

```
func testSendBezierSegment(){
 var testSessionContainer = SessionContainerMock(displayName: "testPlayer")
 var testBezier = BezierSegment(startPoint: CGPoint(x: 0.0, y: 0.0),
  controlPoint1: CGPoint(x: 0.0, y: 0.0), controlPoint2: CGPoint(x: 0.0, y: 0.0),
  endPoint: CGPoint(x: 0.0, y: 0.0), feltPenType: FeltPenType.fine,
  feltPenColor: FeltPenColor.red)
 testSessionContainer.sendBezierSegment(testBezier)
 var testBezierReturnData:NSData = testSessionContainer.getLastSentData()
 var testBezierString = testSessionContainer.prepareMessageWithType(
  MessageType.BezierSegment, message: testBezier.getBezierString())
 var messageData = testBezierString.dataUsingEncoding(NSUTF8StringEncoding, allowLossyConversion: false)
 XCTAssertTrue(messageData == testBezierReturnData, "BezierSegment␣message")
}
```

Listing 4.3: Bezier segment message construction test with Mock

After sending the message with the session mock, the injected method "getLastSentData()" returns the message data. This message data can be compared with referential data to test message integrity. This behavior could also be implemented by delegate callbacks. To handle delegate callbacks in xCode tests, asynchronous test methods had to be implemented.

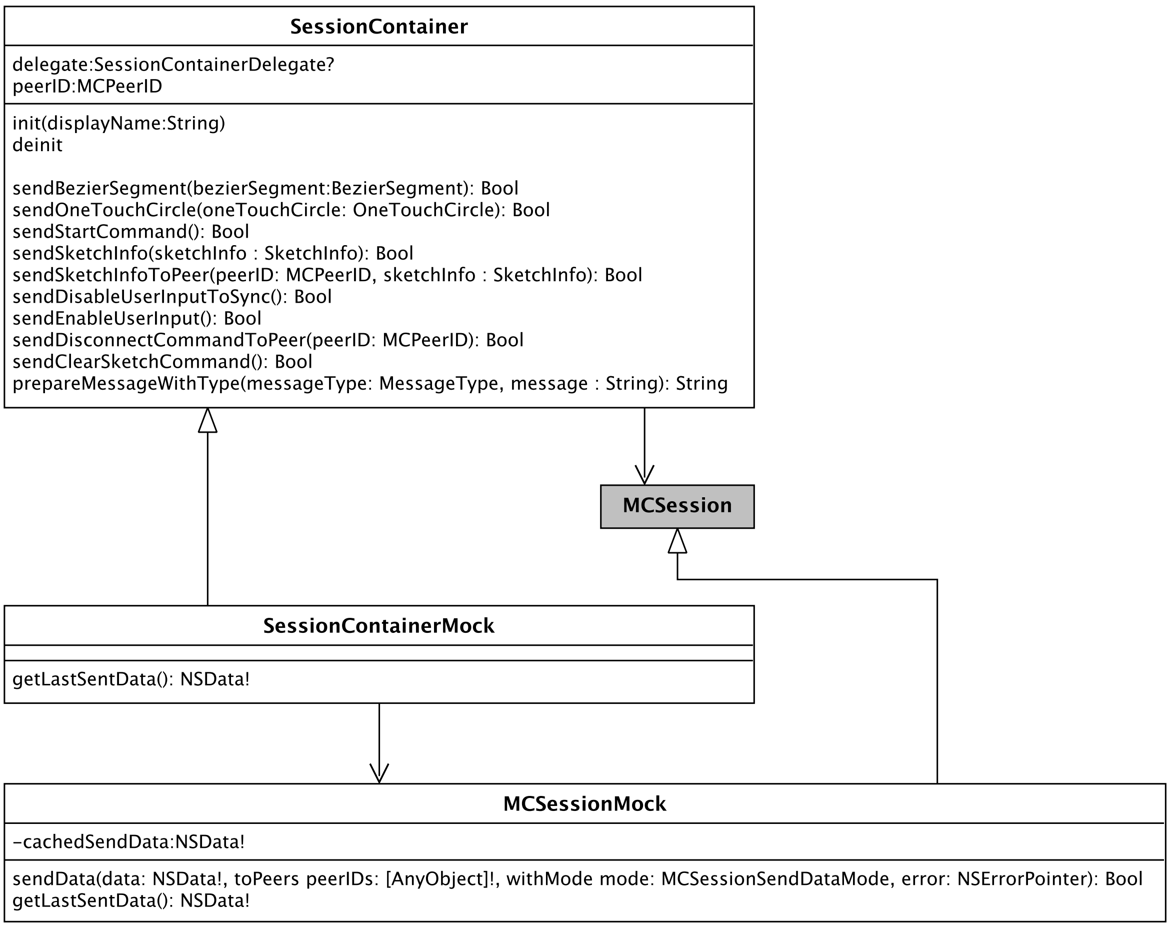

SessionContainer
delegate:SessionContainerDelegate?
peerID:MCPeerID
init(displayName:String)
deinit
sendBezierSegment(bezierSegment:BezierSegment): Bool
sendOneTouchCircle(oneTouchCircle: OneTouchCircle): Bool
sendStartCommand(): Bool
sendSketchInfo(sketchInfo : SketchInfo): Bool
sendSketchInfoToPeer(peerID: MCPeerID, sketchInfo : SketchInfo): Bool
sendDisableUserInputToSync(): Bool
sendEnableUserInput(): Bool
sendDisconnectCommandToPeer(peerID: MCPeerID): Bool
sendClearSketchCommand(): Bool
prepareMessageWithType(messageType: MessageType, message : String): String
MCSession
SessionContainerMock
getLastSentData(): NSData!
MCSessionMock
-cachedSendData:NSData!
sendData(data: NSData!, toPeers peerIDs: [AnyObject]!, withMode mode: MCSessionSendDataMode, error: NSErrorPointer): Bool
getLastSentData(): NSData!

Figure 4.42: Class changes to test message construction

4.9.2 Asynchronous Tests

This test group is used for URLSession tests as well as delegate callback tests. The xCode test automation framework uses the Expectation class (XCTestExpectation) to provide asynchronous test functionality. The TeamsketchSOAP class was tested by implementing its delegate within the test class. Code listing 4.4 shows an asynchronous test example.

```
func testLogin() {
 // ...
 var teamsketchSOAP = TeamsketchSOAP()
 teamsketchSOAP.delegate = self
 teamsketchSOAP.login(userName, password: testPassword)
 waitForExpectationsWithTimeout(20, handler: { error in
  })
}
// ...
func didUserLogin(user: User, password : String){
 XCTAssertNotNil(user, "user should not be nil")
 XCTAssertNotNil(password, "password should not be nil")
 // ...
 XCTAssertTrue((user.userName == userName) &&
  (password == testPassword), "loginData")
 expectation.fulfill()
}
```

Listing 4.4: Asynchronous test implementation example

5 Web Service

Teamsketch should be used in classrooms to train collaborative and teamwork of primary school students. An online image store and receive service was implemented to access the images on other devices. Data should be available on iPads and PCs as well as other capable devices. The SOC[1] paradigm provides these desired requirements [Papazoglou et al., 2007, p. 38]. Teamsketch web service offers a device independent API via self-describing XML requests and responses. Web services offer a well tested environment to support distributed information processing. However, one drawback of this solution is the decreased system performance [Chiu, Govindaraju, and Bramley, 2002, p. 2]. This drawback is not a significant issue as image upload and download speed is not critical for the Teamsketch functionality.

5.1 Features

The following requirements were defined:

- students can authenticate against University Usermanagement SOAP[2]
- students can upload their sketches to the web service
- images should be stored in two different sizes (original size and thumbnail)
- four images for each user are stored (one profile picture and three sketches). Adapting the web service to other values should be easy to implement.
- web service should provide an API for web applications.

[1]Service Oriented Computing
[2]`http://mathe.tugraz.at/~georg/Usermanager/public/soap?wsdl`, last accessed: 12. Nov. 2014

Table 5.1: Web service requests and descriptions

request	description
login	authentication with user name and password.
getImage	returns a specific image
getImageSizes	returns the supported number of image sizes
getNumberOfUserImages	returns the number of images which can be stored by a user
setImage	uploads an image to the web service

5.2 System Architecture

The web service is implemented in PHP[3] as a SOAP service defined in WSDL[4]. These technologies were not only chosen because of the already existing infrastructure but also because of the possibility to use well known message transmission with HTTP requests and responses. The already existing UserManager, which handles authentication, uses the same infrastructure. Table 5.1 gives an overview about the used requests.

The Teamsketch web service interacts with the UserManagement SOAP. Figure 5.1 elaborates on the communication details.

5.2.1 SOAP

SOAP describes an XML based message format used to communicate with web services embedded in arbitrary transport protocols. To access data from the web service a request must be defined and translated into a SOAP message. The server extract the SOAP message and processes the information. A SOAP message is an XML document which consists of the following parts.

- SOAP envelope
- SOAP header
- SOAP body

[3]PHP: Hypertext Preprocessor
[4]Web Services Description Language

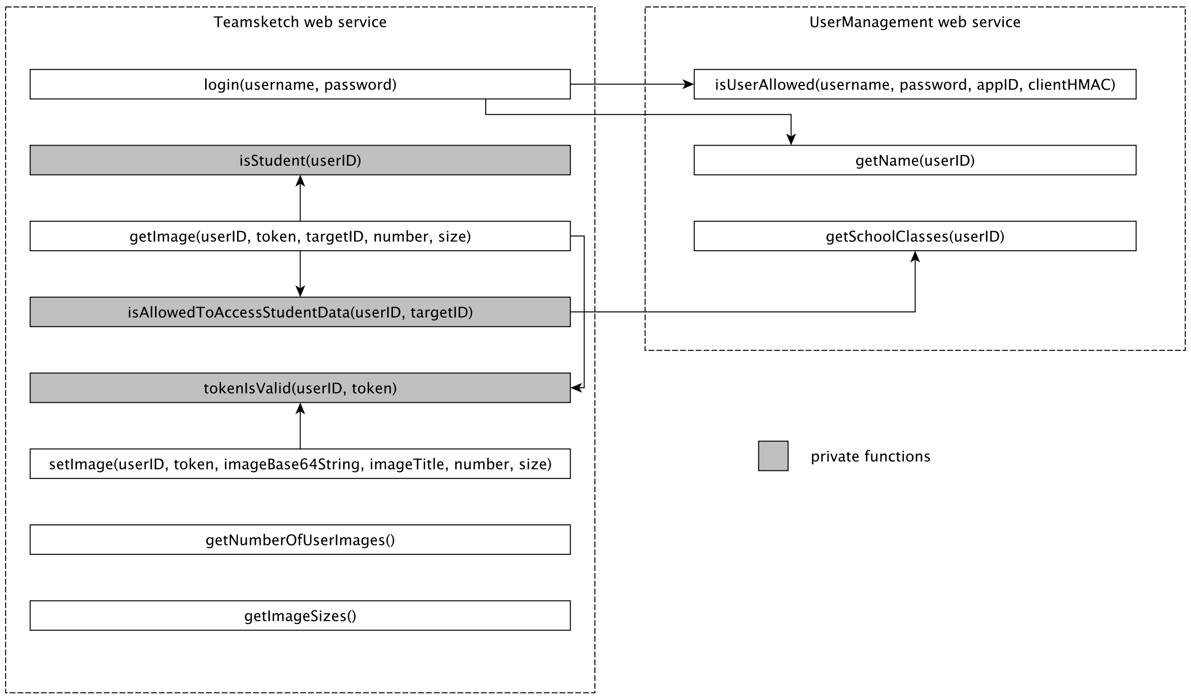

Teamsketch web service
login(username, password)
isStudent(userID)
getImage(userID, token, targetID, number, size)
isAllowedToAccessStudentData(userID, targetID)
tokenIsValid(userID, token)
setImage(userID, token, imageBase64String, imageTitle, number, size)
getNumberOfUserImages()
getImageSizes()
UserManagement web service
isUserAllowed(username, password, appID, clientHMAC)
getName(userID)
getSchoolClasses(userID)
private functions

Figure 5.1: Web service communication

The SOAP envelope is the root of the XML document which defines the SOAP type and other service related parameters. The SOAP header is optional, it is often used to transmit security related information. The SOAP body contains the actual request information [Melzer, 2010, p. 87-91]. Listing 5.1 shows the SOAP envelope of the setImage request. The question marks represents the request parameters set by the user.

```
<soapenv:Envelope
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
          xmlns:soap="http://schule.learninglab.tugraz.at/teamsketch/soap">
   <soapenv:Header/>
   <soapenv:Body>
      <soap:setImage
         soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
          <userID xsi:type="xsd:int">?</userID>
          <token xsi:type="xsd:string">?</token>
          <imageBase64 xsi:type="xsd:string">?</imageBase64>
          <imageTitle xsi:type="xsd:string">?</imageTitle>
          <number xsi:type="xsd:int">?</number>
          <size xsi:type="xsd:int">?</size>
      </soap:setImage>
   </soapenv:Body>
</soapenv:Envelope>
```

Listing 5.1: setImage request SOAP envelope

5.2.2 WSDL

WSDL is used to describe the SOAP message format and the SOAP interface in detail. The WSDL information is specified in XML and is separated in various sections [Christensen et al., 2001]:

- definition - defines several environment parameters
- documentation - adds human readable documentation
- types - defines message response objects and their properties (names and types)
- message - defines SOAP message information such as message parameter names and types as well as message response information
- portType - describes the request name connected with the corresponding message (request and response)
- binding - defines how to access the service
- service - describes where the service is located (URI[5])

Listing 5.2 describes the WSDL code used to define the setImage request.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
             xmlns:tns="http://schule.learninglab.tugraz.at/teamsketch/soap"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:soap-enc="http://schemas.xmlsoap.org/soap/encoding/"
             xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
             name="Teamsketch"
             targetNamespace="http://schule.learninglab.tugraz.at/teamsketch/soap">

<!-- ... -->
<message name="setImageIn">
        <part name="userID" type="xsd:int"/>
        <part name="token" type="xsd:string"/>
        <part name="imageBase64" type="xsd:string"/>
        <part name="imageTitle" type="xsd:string"/>
        <part name="number" type="xsd:int"/>
        <part name="size" type="xsd:int"/>
</message>
<message name="setImageOut">
        <part name="return" type="xsd:boolean"/>
</message>
<!-- ... -->
<portType name="TeamsketchPort">
        <!-- ... -->
        <operation name="setImage">
                <documentation>This Method sets an image from base64 encoded string</documentation>
                <input message="tns:setImageIn"/>
                <output message="tns:setImageOut"/>
        </operation>
        <!-- ... -->
</portType>
<!-- ... -->
<binding name="TeamsketchBinding" type="tns:TeamsketchPort">
        <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <!-- ... -->
    <operation name="setImage">
        <soap:operation soapAction="http://schule.learninglab.tugraz.at/teamsketch/soap/#setImage"/>
        <input>
```

[5]Uniform Resource Identifier

```
                <soap:body use="encoded"
                        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                        namespace="http://schule.learninglab.tugraz.at/teamsketch/soap"/>
        </input>
        <output>
                <soap:body use="encoded"
                        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                        namespace="http://schule.learninglab.tugraz.at/teamsketch/soap"/>
        </output>
        </operation>
        <!-- ... -->
</binding>
<service name="TeamsketchService">
        <port name="TeamsketchPort" binding="tns:TeamsketchBinding">
                <soap:address location="http://schule.learninglab.tugraz.at/teamsketch/soap/"/>
        </port>
</service>
</definitions>
```

Listing 5.2: setImage WSDL information

5.2.3 Communication

The web service communicates with requests and responses built in XML. The functionality of Teamsketch requests and response will be demonstrated by the ”getImage” procedure. The next section describes the HTTP information transmission based on the ”getImage” request. User relevant data is substituted with ”...”.

Request

All requests use a HTTP POST request. Table 5.2 describes the POST parameters and listing 5.3 shows the ”getImage” POST body details.

Table 5.2: getImage HTTP POST header

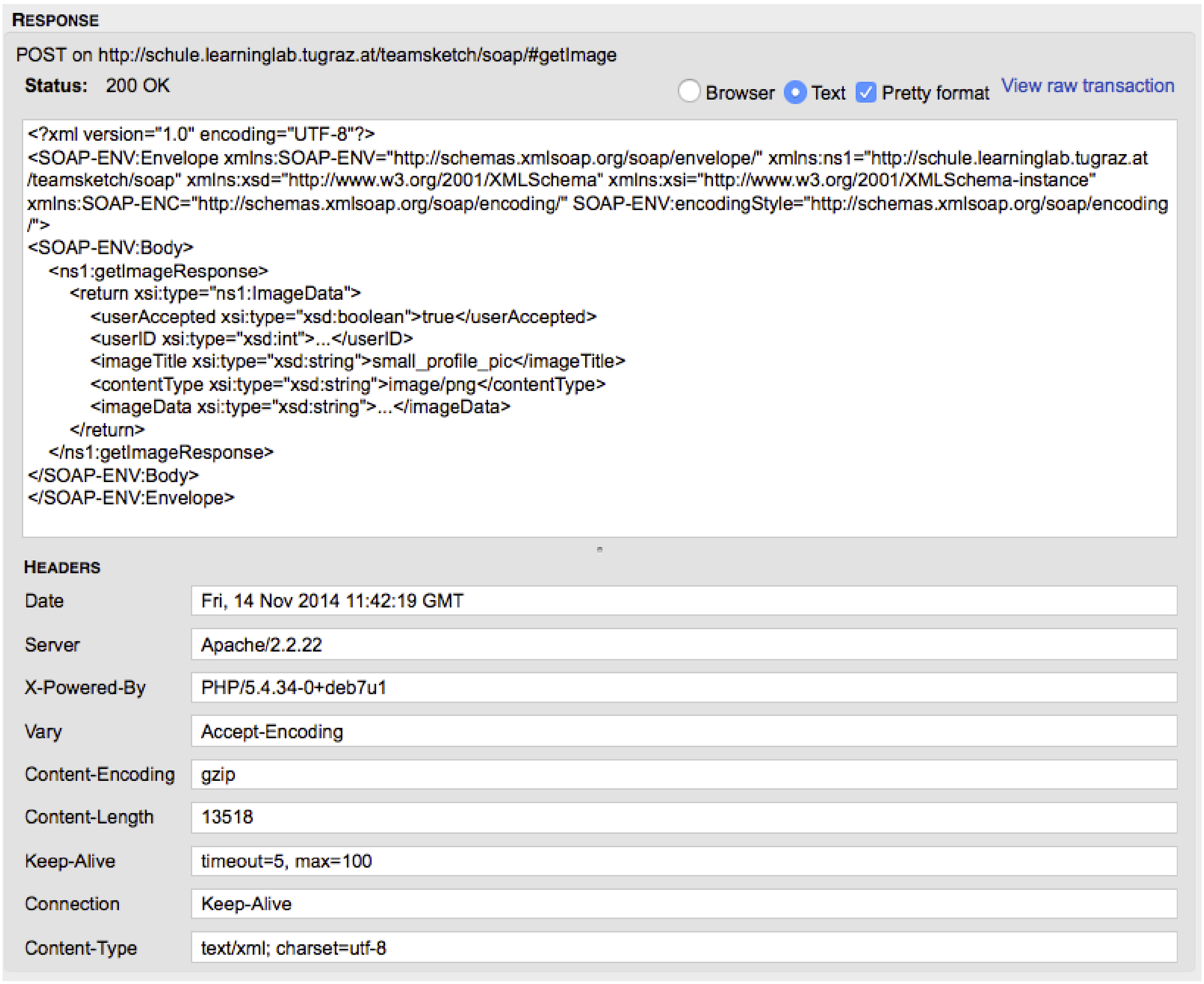
Name	Value
Accept-Encoding	gzip,deflate
Content-Type	text/xml;charset=UTF-8
SOAPAction	http://schule.learninglab.tugraz.at/teamsketch/soap/#getImage
Connection	Keep-Alive

```
  <soapenv:Header/>
   <soapenv:Body>
      <soap:getImage soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <userID xsi:type="xsd:int">...</userID>
         <token xsi:type="xsd:string">...</token>
         <targetID xsi:type="xsd:int">...</targetID>
         <number xsi:type="xsd:int">0</number>
         <size xsi:type="xsd:int">0</size>
      </soap:getImage>
   </soapenv:Body>
</soapenv:Envelope>
```

Listing 5.3: getImage HTTP POST body

Response

The result of the request will be sent as HTTP response. Figure 5.2 shows getImage response details.


RESPONSE

POST on http://schule.learninglab.tugraz.at/teamsketch/soap/#getImage

Status: 200 OK

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns1="http://schule.learninglab.tugraz.at
/teamsketch/soap" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding
/">
<SOAP-ENV:Body>
  <ns1:getImageResponse>
    <return xsi:type="ns1:ImageData">
      <userAccepted xsi:type="xsd:boolean">true</userAccepted>
      <userID xsi:type="xsd:int">...</userID>
      <imageTitle xsi:type="xsd:string">small_profile_pic</imageTitle>
      <contentType xsi:type="xsd:string">image/png</contentType>
      <imageData xsi:type="xsd:string">...</imageData>
    </return>
  </ns1:getImageResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

HEADERS

Date	Fri, 14 Nov 2014 11:42:19 GMT
Server	Apache/2.2.22
X-Powered-By	PHP/5.4.34-0+deb7u1
Vary	Accept-Encoding
Content-Encoding	gzip
Content-Length	13518
Keep-Alive	timeout=5, max=100
Connection	Keep-Alive
Content-Type	text/xml; charset=utf-8

Figure 5.2: getImage HTTP response

5.2.4 Image Data Transmission

Two different techniques were considered during the implementation of the image data transfer. One method uses base64 encoded strings to transport image information which is embedded in the XML POST body. This technique is easy to implement but has some drawbacks such as bloating the SOAP message. Furthermore, the receiver must know the file type of the transmitted data to decode the transmitted information. Another approach is to sent binary data with MTOM[6]. One significant advantage

[6]Message Transmission Optimization Mechanism

of MTOM is that files can be set without further encoding. This optimizes the data transfer due to less size of the XML file [Mhatre, Mehta, and Jaiswal, 2013, p. 1]. The images transferred within Teamsketch are not big in size therefore the image is embedded as base64 encoded string in the response XML file. The other main reason why base64 encoding was used is that the consuming web interface needs base64 encoded data to embed images in HTML[7]. Additionally, the iPad app uses base64 encoded data to send images.

5.3 Design Patterns

The web service uses the MVC architecture as well as other previously described patterns such as Facade pattern.

5.4 User Interface

The web service user interface is only accessible by SOAP requests. No GUI[8] or other user interfaces are provided. The user interface for pupils and teachers is implemented by a web interface which will be explained in chapter 6.

5.5 Implementation Details

Teamsketch was implemented with Netbeans 8.0.1. The service is running on a Linux server with PHP version 5.4.34-0deb7u1.

5.5.1 Overview

The following components are involved:

- Teamsketch - implements all public SOAP service methods as well as some private helper messages
- User - holds user information from login response
- Image - holds image information from getImage response

Figure 5.3 elaborates on the details of Teamsketch web service.

[7]HyperText Markup Language
[8]Graphical User Interface

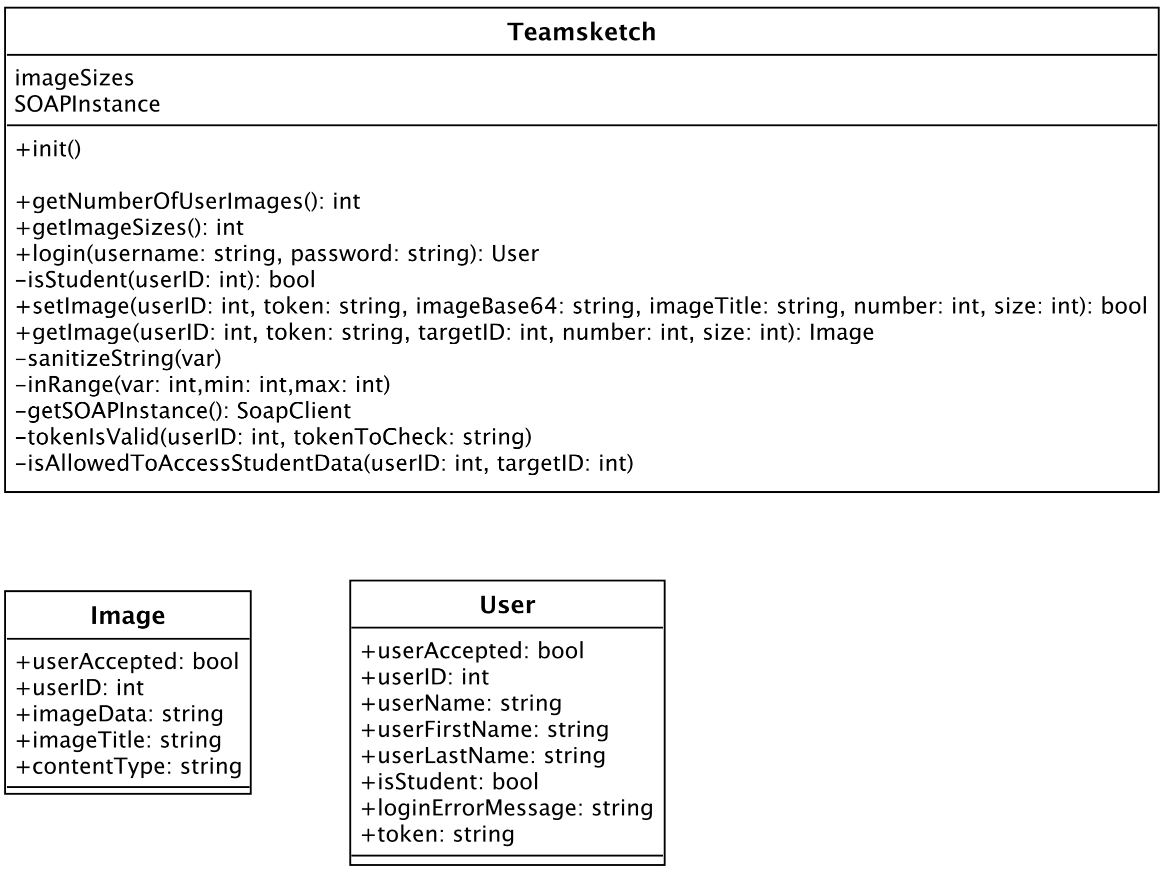

Teamsketch
imageSizes
SOAPInstance
+init()
+getNumberOfUserImages(): int
+getImageSizes(): int
+login(username: string, password: string): User
−isStudent(userID: int): bool
+setImage(userID: int, token: string, imageBase64: string, imageTitle: string, number: int, size: int): bool
+getImage(userID: int, token: string, targetID: int, number: int, size: int): Image
−sanitizeString(var)
−inRange(var: int,min: int,max: int)
−getSOAPInstance(): SoapClient
−tokenIsValid(userID: int, tokenToCheck: string)
−isAllowedToAccessStudentData(userID: int, targetID: int)
Image
+userAccepted: bool
+userID: int
+imageData: string
+imageTitle: string
+contentType: string
User
+userAccepted: bool
+userID: int
+userName: string
+userFirstName: string
+userLastName: string
+isStudent: bool
+loginErrorMessage: string
+token: string

Figure 5.3: Web service overview

5.5.2 SOAP server initialization

All SOAP requests are implemented within the Teamsketch class. Listing 5.4 shows the code which was used to activate the SOAP server. The SOAP server is initialized with the corresponding Teamsketch WSDL file. Then the Teamsketch web service class is assigned. The service starts with the "handle()" command.

```
    $server = new SOAPServer(TEAMSKETCH_WSDL);
    $server->setClass("Teamsketch");
    $server->handle();
```

Listing 5.4: PHP SOAP server initialization

5.5.3 Login

The login request is forwarded to the already existing UserManagement SOAP service. If the user authentication fails, an empty User object will be returned which holds the login error message. When access is granted, a user object with all available user information such as user name and ID will be generated. The Teamsketch web service calculates a token which will be returned as well. With this token Teamsketch image upload and download is possible without further authentication against UserManagement. The reason why tokens are used is that UserManagement service requests are slow due to the service chain (login request - Teamsketch SOAP - User Management SOAP).

5.5.4 getImage

The public function getImage needs five parameters:

- userID
- token
- targetID
  This is the ID of the image owner. Some users (teachers) can access images of other users (pupils).
- number - image number (0: profile image, 1-3: sketches)
- size - 0: thumbnail, 1: original size

Before further processing, the access token has to be checked. If the token is valid and number as well as size is within an appropriate range, the user role will be investigated. If the user is a pupil, only own sketches can be accessed (userID == targetID). If the user is a teacher the system checks if the targetID belongs to a pupil of the teacher's classes. If this is the case, the teacher is allowed to access the sketches. Figure 5.4 elaborates on the processing details of this function.

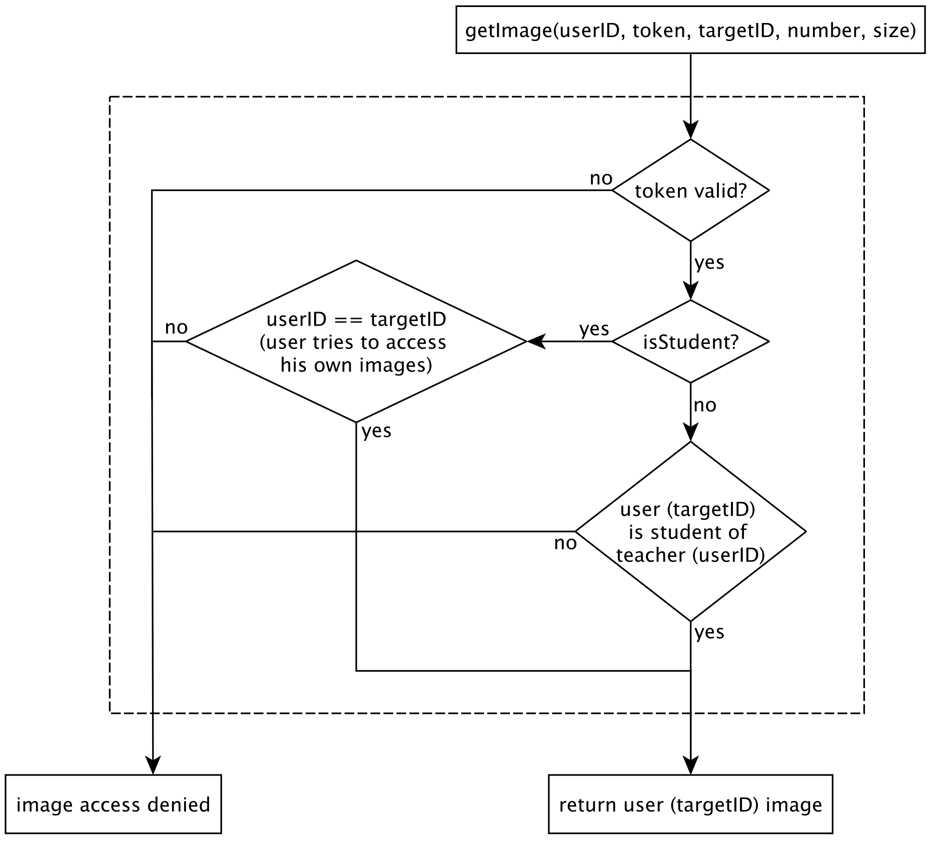

getImage(userID, token, targetID, number, size)
token valid?
no
yes
isStudent?
yes
userID == targetID (user tries to access his own images)
no
yes
no
user (targetID) is student of teacher (userID)
no
yes
image access denied
return user (targetID) image

Figure 5.4: getImage information processing

5.5.5 setImage

The public function setImage needs six parameters:

- userID
- token
- imageBase64 - image as base64 encoded string

- imageTitle - user defined image title
- number - image number
- size - image size

If the token is valid and size as well as number are appropriate , the image will be stored.

5.6 Unit Testing

SoapUI[9] was used to test the SOAP service. It supports importing WSDL files and automatically creates example requests. The suite supports functional testing as well[10].

5.6.1 Functional Testing

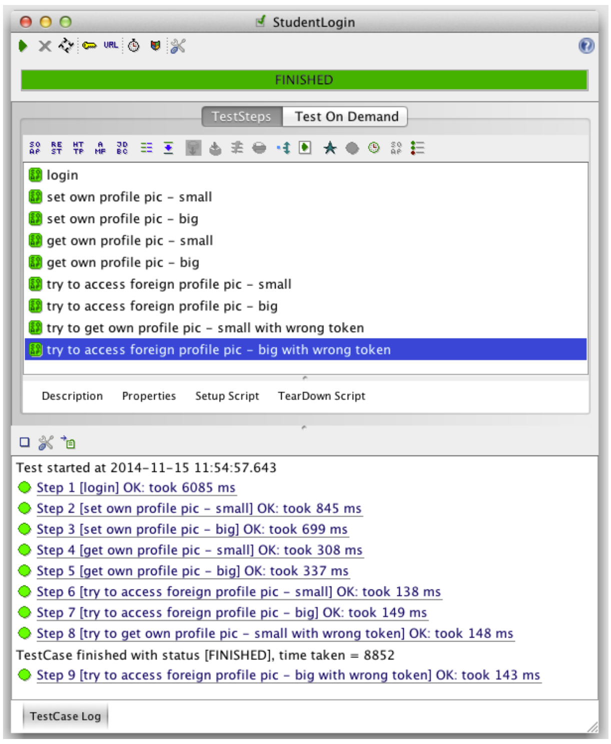
To test web service functionality, many functional test have been added. The following figure 5.5 shows details of the login function tests. Figure 5.6 shows one test case of the login test suite.

Every test step verifies three assertions:

- SOAP response valid - the web service must return a valid SOAP response
- not SOAP fault - no SOAP fault response happened
- XPath Match - the response is compared to a previously defined XML response

[9] `http://www.soapui.org/`, last accessed: 15. Nov. 2014

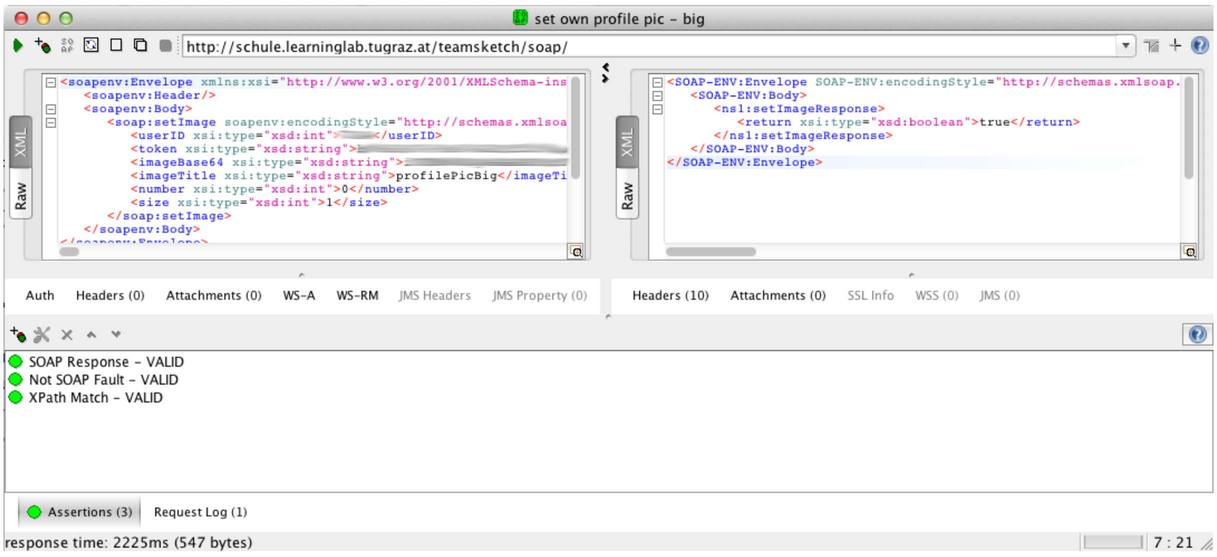
[10] `http://www.soapui.org/Functional-Testing/functional-testing.html`, last accessed: 15. Nov. 2014



```
login
set own profile pic – small
set own profile pic – big
get own profile pic – small
get own profile pic – big
try to access foreign profile pic – small
try to access foreign profile pic – big
try to get own profile pic – small with wrong token
try to access foreign profile pic – big with wrong token
```

```
Test started at 2014-11-15 11:54:57.643
Step 1 [login] OK: took 6085 ms
Step 2 [set own profile pic – small] OK: took 845 ms
Step 3 [set own profile pic – big] OK: took 699 ms
Step 4 [get own profile pic – small] OK: took 308 ms
Step 5 [get own profile pic – big] OK: took 337 ms
Step 6 [try to access foreign profile pic – small] OK: took 138 ms
Step 7 [try to access foreign profile pic – big] OK: took 149 ms
Step 8 [try to get own profile pic – small with wrong token] OK: took 148 ms
TestCase finished with status [FINISHED], time taken = 8852
Step 9 [try to access foreign profile pic – big with wrong token] OK: took 143 ms
```

Figure 5.5: Web service login test suite



```
http://schule.learninglab.tugraz.at/teamsketch/soap/
```

```xml
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-ins
   <soapenv:Header/>
   <soapenv:Body>
      <soap:setImage soapenv:encodingStyle="http://schemas.xmlsoa
         <userID xsi:type="xsd:int">[redacted]</userID>
         <token xsi:type="xsd:string">[redacted]
         <imageBase64 xsi:type="xsd:string">[redacted]
         <imageTitle xsi:type="xsd:string">profilePicBig</imageTi
         <number xsi:type="xsd:int">0</number>
         <size xsi:type="xsd:int">1</size>
      </soap:setImage>
   </soapenv:Body>
```

```xml
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.
   <SOAP-ENV:Body>
      <ns1:setImageResponse>
         <return xsi:type="xsd:boolean">true</return>
      </ns1:setImageResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
SOAP Response – VALID
Not SOAP Fault – VALID
XPath Match – VALID
```

```
response time: 2225ms (547 bytes)
```

Figure 5.6: Web service login test suite - test example

6 Web Interface

Teamsketch not only provides a web service to store and retrieve image data but also offers a web interface for accessing images implemented for teachers and pupils.

6.1 Features

The following requirements were defined:

- Only authorized users (teacher and students) should be allowed to access image data.
- separate interfaces for students and teachers
- pupils should be allowed to see only their own images
- teachers should be allowed to see only images of their own students
- teachers can browse their classes for students and images
- images should be displayed on screen as thumbnails
- download option for images in full resolution
- multi language support (English and German)

Additionally, some technical requirements had to be fulfilled:

- web interface can only access user and image data through public UserManagement API and Teamsketch API
- host independent implementation
- no absolute paths allowed
  If absolute paths are not avoidable they have to be declared in an easy locatable configuration file.

- no JavaScript
  In schools the computer systems as well as browsers are usually not up to date. For maximum compatibility newer technologies should be avoided.

6.2 Design patterns

The web interface follows the MVC pattern. The data model is provided by UserManagement API and Teamsketch API. The controllers implement session management and web service work flow. The framework Smarty[1] was used to implement the view part.

6.3 System Architecture

The web interface is implemented in PHP. The visualization is done with HTML and CSS[2]. Table 6.1 shows used frameworks.

Table 6.1: Web interface frameworks

Framework	Version
PHP	5.4.34-0deb7u1
Smarty	3.1-2.19

The web interface is separated into different parts:

- index.php
  Displays the welcome page.

- login.php
  Manages the user login and forwards the user to "teacher.php" or "student.php" depending on the user role.

- student.php
  Displays student images (profile picture as well as sketches).

- teacher.php
  Displays the teacher interface and provide access to the teacher's classes.

[1]http://www.smarty.net/, last accessed: 15. Nov. 2014
[2]Cascading Style Sheets

- class.php
  Provides access to student of a certain class.

6.3.1 Smarty

The Teamsketch web interface uses Smarty to generate HTML responses. Smarty provides a template engine for PHP. With this framework, content and program logic can be separated easily to follow the MVC pattern. Each HTML visualization is stored in a template file. Within these files static HTML commands are used as well as variable content which can be changed by PHP. Smarty was used because of the following features[3]:

- fast
- cache handling
- will recompile only if template files are changed

6.3.2 Web Service Communication

The web interface uses public API functions to access image and user information. Figure 6.1 shows involved API calls.

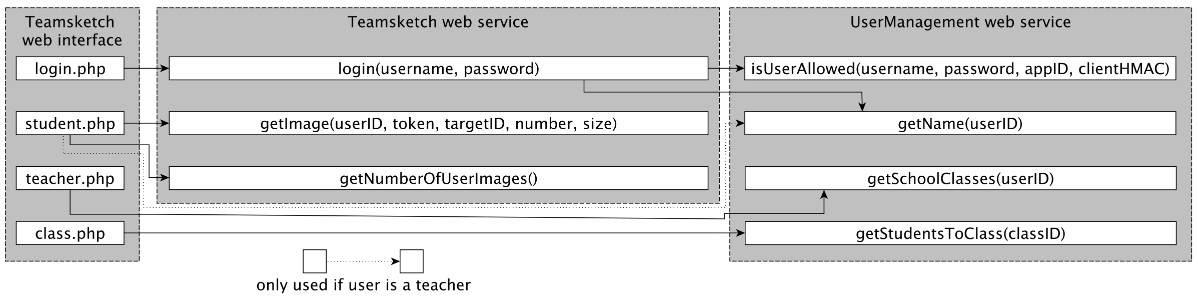

Teamsketch web interface
login.php
student.php
teacher.php
class.php
Teamsketch web service
login(username, password)
getImage(userID, token, targetID, number, size)
getNumberOfUserImages()
UserManagement web service
isUserAllowed(username, password, appID, clientHMAC)
getName(userID)
getSchoolClasses(userID)
getStudentsToClass(classID)
only used if user is a teacher

Figure 6.1: Web interface API call overview

6.4 User Interface

The user interface depends on the user role. The web service starts by displaying a welcome page where users are asked to input their user credentials. After access is

[3] http://www.smarty.net/docsv2/en/what.is.smarty.tpl, last accessed: 15. Nov. 2014

granted the user role will be investigated. If the user is a pupil, the next view will be the student view. In this view, users can see the thumbnails of their user image and their user sketches. If teachers use the web service, a list of their classes will be displayed. When a class is selected, all students of this class will be listed. After clicking on a pupil's name, the pupil's images can be accessed. Figure 6.2 elaborates on the details of the view work flow.

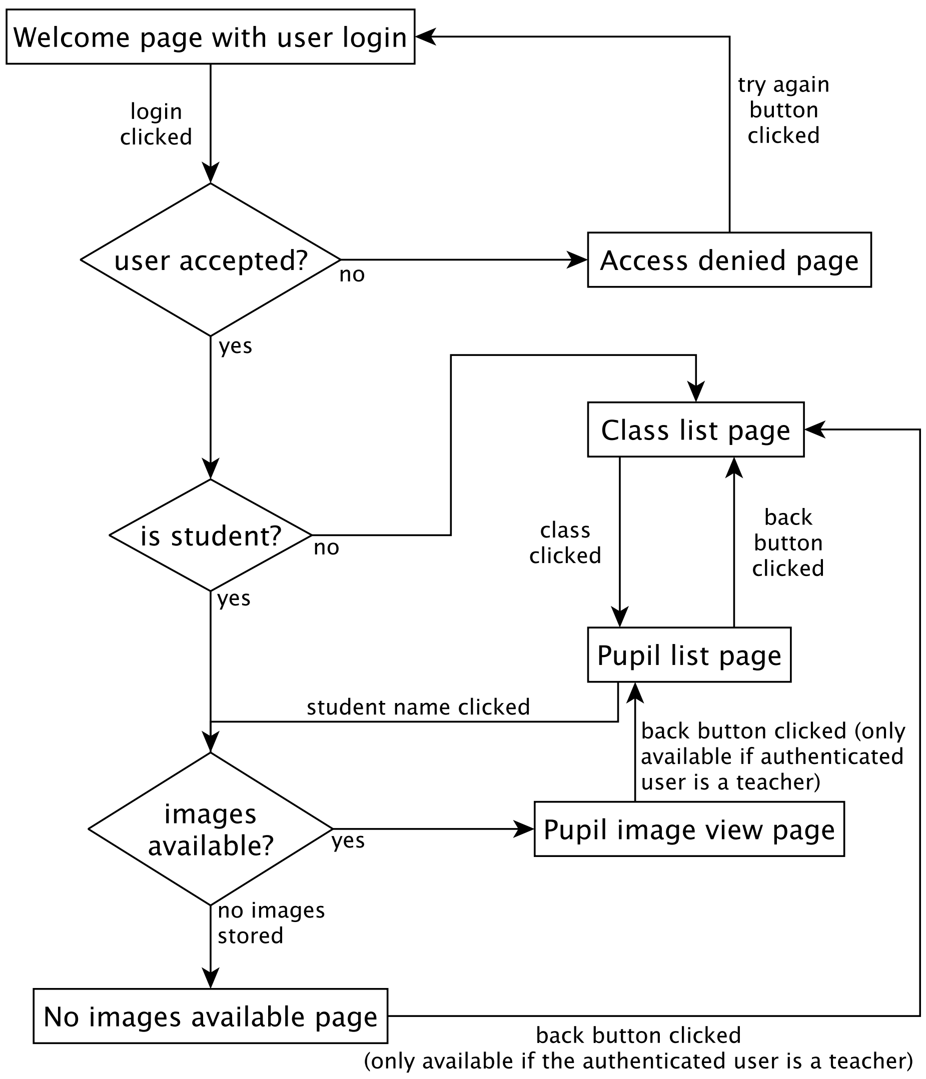


Figure 6.2: User interface overview

6.4.1 Welcome page

This page is the start page of the web interface. User can enter their login credentials or create a new account. Figure 6.3 shows the welcome page.

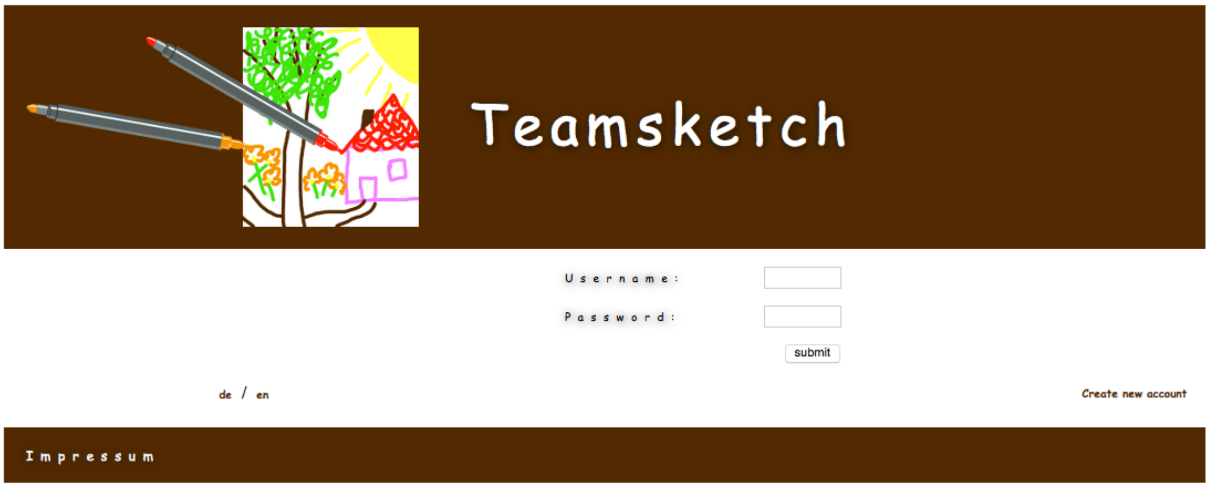


Figure 6.3: Welcome page

6.4.2 Student Interface

After the user logged in successfully, the user is a pupil and user images are available, the student image view will be shown. Figure 6.4 shows the student image page. The first image contains a thumbnail of the user profile picture. The section ”Sketches” shows thumbnails of all user sketches grouped to three in a row. User sketches can be downloaded in full resolution. Teamsketch offers three sketch slots in the moment but this could be changed easily by adapting the image count constant within the Teamsketch web service.

6.4.3 Teacher Interface

When a user was identified as a teacher, a class list will be shown (Figure 6.5). Teacher can click on the class names to open the ”students in class” view.

Students in Class

When the teacher clicks on a class, the result page lists all students of the selected class (Figure 6.6).

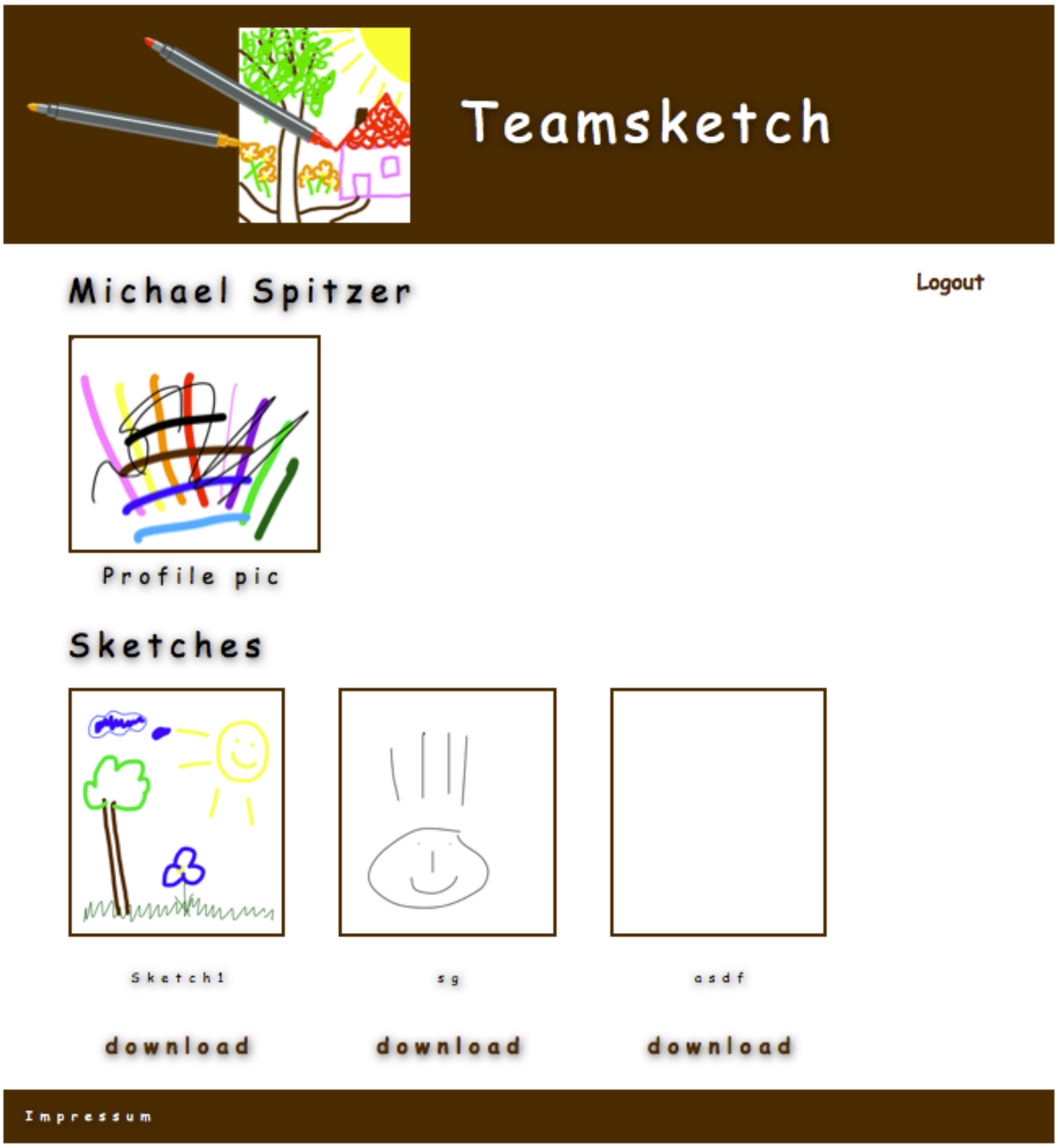
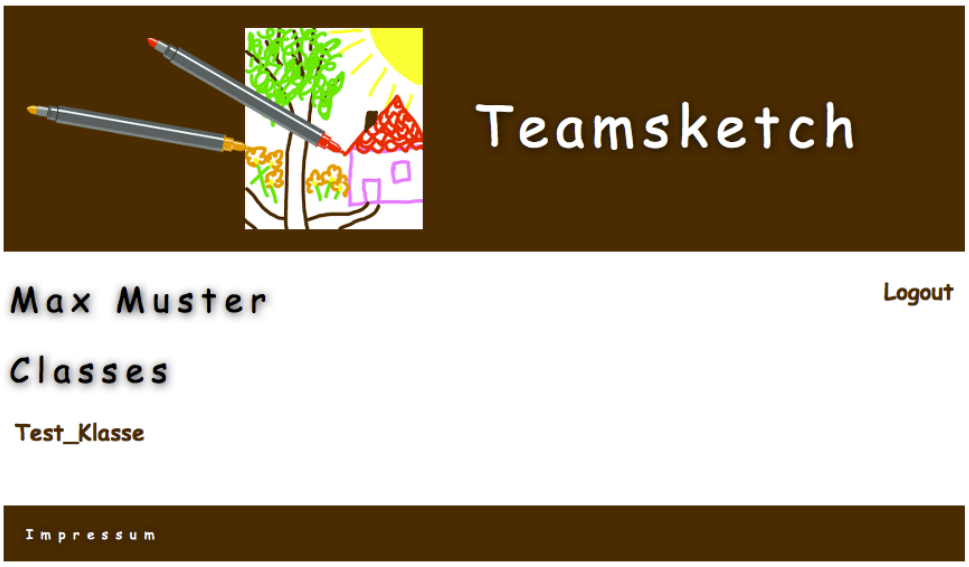


Figure 6.4: Web interface pupil image page


Teamsketch

Max Muster

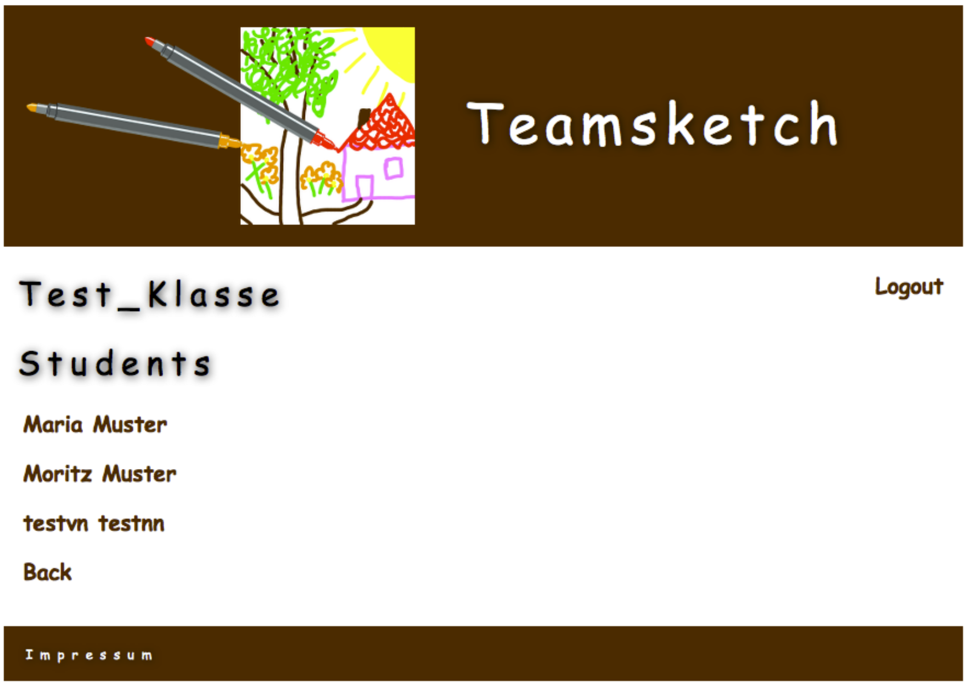
Logout

Classes

Test_Klasse

Impressum

Figure 6.5: Web interface teacher's class list page


Teamsketch

Test_Klasse

Logout

Students

Maria Muster

Moritz Muster

testvn testnn

Back

Impressum

Figure 6.6: Students in class page

6.5 Implementation Details

The web interface was implemented in a Vagrant[4] development environment. Vagrant uses a Virtualbox[5] image with SSH[6] access to test web services and applications in a virtual environment. The image was an exact copy of the Teamsketch production server (Linux version, installed libraries and active extensions). Therefore the web interface could be tested in production server environment without putting the real server at risk while testing.

6.5.1 Session management

After the user logged in successfully, a session is created. Figure 6.7 shows the stored session variables and details as well as the transmitted GET parameters for teachers. If German is selected, every link uses the language GET parameter. Therefore the language GET parameter is not mentioned in the figure. Session management for pupils is trivial because there is only one single page available for them.

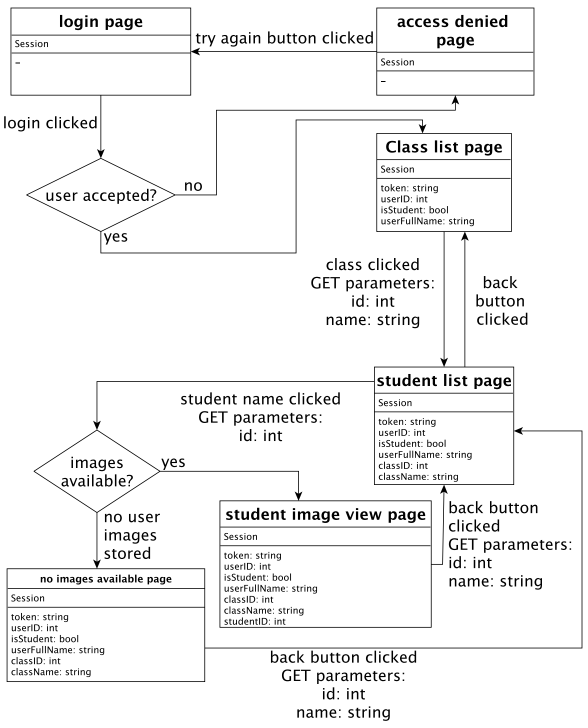


Figure 6.7: Session details for teachers

[4] https://www.vagrantup.com, last accessed: 15. Nov. 2014
[5] https://www.virtualbox.org, last accessed: 15. Nov. 2014
[6] Secure SHell

6.5.2 Template engine

The template engine simplifies managing dynamic web content. The web service (current implementation) returns three user sketches for each user. What if the web service will be improved that it can handle more than three user images? In this case the template engine provides features to simplify this issue. Listing 6.1 shows the PHP part of the code generation for the sketch images. Listing 6.2 elaborates on Smarty template details.
At first the web service is queried to get the number of sketch images stored for each user. The second step is to retrieve all sketch image thumbnails as base64 encoded string as well as the image title. Smarty supports array parameters, therefore this information is forwarded with arrays (sketches and sketchTitles). The template code consists of HTML tags (static code) and Smarty commands (in curly brackets). Images are nested in sections. These sections support looping through an array and create code blocks for each element of the array. The images are grouped in three images in each line. Smarty also supports if-then-else commands to implement conditioned HTML code insertion. This is needed to stop image insertion if image array reaches the last element.

```
$numberOfSketches = TeamsketchSOAPClient::getSOAPClient()->getNumberOfUserImages() - 1; // without profile pic;
for($i = 0;$i<$numberOfSketches;$i++){
        $currentSketch = TeamsketchSOAPClient::getSOAPClient()->getImage($userID, $token, $studentID,$i+1,0);
        $sketches[$i] = $currentSketch->imageData;
        $sketchTitles[$i] = $currentSketch->imageTitle;
}
$smarty->assign('sketchArray', $sketches);
$smarty->assign('sketchTitleArray', $sketchTitles);
```

Listing 6.1: Sketch image code generation

```
{section name=sketchNumber loop=$sketchArray}
 {if $smarty.section.sketchNumber.index%3 === 0}
  <div class="sketches">
   {if $smarty.section.sketchNumber.index lt $sketchArray|@count}
    {assign var="currentSketchNumber" value=$smarty.section.sketchNumber.index}
    <div id="sketch_left">
     {html_image file="data:image/png;base64,{$sketchArray[{$currentSketchNumber}]}" height="222" width="192"}
     <div class="sketch_caption"><p>{$sketchTitleArray[{$currentSketchNumber}]}</p>
      <a class ="link" href="io/imageDownload.php?id={$currentSketchNumber+1}">download</a>
     </div>
    </div>
   {/if}
   {if $smarty.section.sketchNumber.index+1 lt $sketchArray|@count}
    {assign var="currentSketchNumber" value=$smarty.section.sketchNumber.index+1}
     <div id="sketch_middle">
      {html_image file="data:image/png;base64,{$sketchArray[{$currentSketchNumber}]}" height="222" width="192"}
      <div class="sketch_caption"><p>{$sketchTitleArray[{$currentSketchNumber}]}</p>
       <a class ="link" href="io/imageDownload.php?id={$currentSketchNumber+1}">download</a>
      </div>
     </div>
   {/if}
   {if $smarty.section.sketchNumber.index+2 lt $sketchArray|@count}
    {assign var="currentSketchNumber" value=$smarty.section.sketchNumber.index+2}
    <div id="sketch_right">
     {html_image file="data:image/png;base64,{$sketchArray[{$currentSketchNumber}]}" height="222" width="192"}
     <div class="sketch_caption"><p>{$sketchTitleArray[{$currentSketchNumber}]}</p>
      <a class ="link" href="io/imageDownload.php?id={$currentSketchNumber+1}">download</a>
     </div>
    </div>
   {/if}
  </div>
 {/if}
{/section}
```

Listing 6.2: Sketch image template code

6.5.3 Multi Language Support

When the pages get changed the language is set as a GET parameter. Page templates were implemented in German and English and will be selected on demand.

6.5.4 SOAP Service access

At first the PHP soap client has to be initialized. Then the soap client object can be used to send queries to the web service. Listing 6.3 shows the usage of the PHP SOAP client[7]

```
$soapClient = new SoapClient(TEAMSKETCH_WSDL);
$sketch1 = $soapClient->getImage($userID, $token, $studentID,1,0);
```

Listing 6.3: PHP SOAP client access

6.6 Unit Testing

PHPUnit[8] was used to test the web interface. PHPUnit is an xUnit test suite which supports automatic software testing [Meszaros, 2007, p. 745]. It is based on the same concepts as it has already been elaborated in iOS unit testing.

[7] http://php.net/manual/de/class.soapclient.php, last accessed: 15. Nov. 2014
[8] https://phpunit.de, last accessed: 15. Nov. 2014

7 Prototype Test in a Primary School

Teamsketch was tested in a primary school (third grade). The selected school (VS Hirten-Graz) has a so called ”iPad class”. The pupils are familiar and experienced with iPads because iPads are integrated in the regular lessons since first grade.

7.1 Test parameters

The school class consisted of 16 pupils. The app test was part of the art lesson. Pupils tested the app in groups of four, sitting on one table.

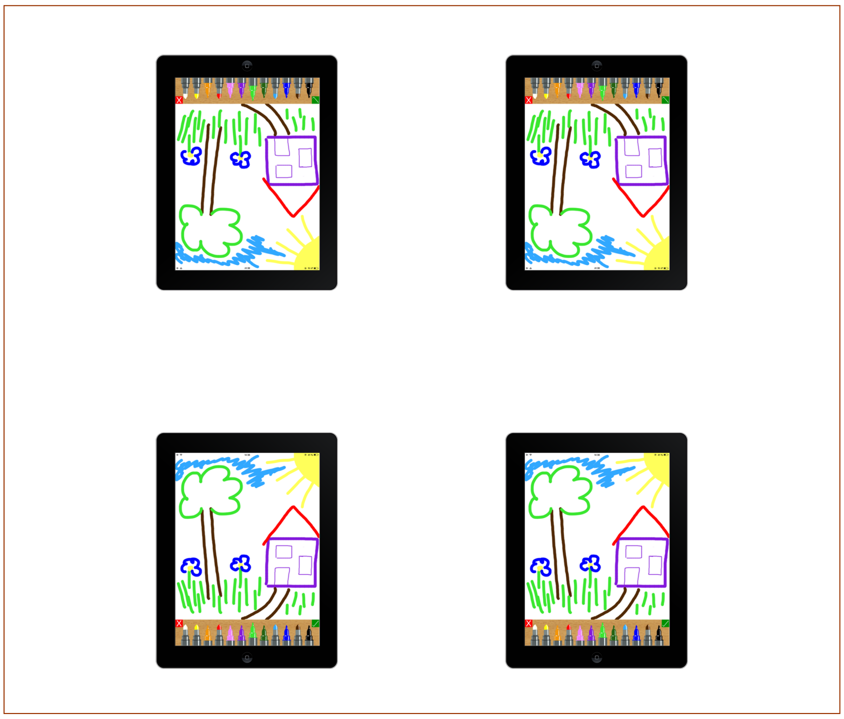


Figure 7.1: Test setting in school class

The first task was to connect the iPads over Wi-Fi. The pupils were already familiar with pairing the devices since they have already used other TU-Graz apps such as

Buchstabenpost[1], which use similar pairing techniques.
The next step was to form a group of two. Each team had to decide a leader on their own. After testing the app in pairs, all four pupils drew one sketch together. The last step was to discuss the app by answering predefined questions. This test procedure is based on previous tests [Kienleitner, 2013, p. 79] as the pupils are used to App testing that way. The main goal was to draw a picture with a given topic such as "farm", "city", "flower" and "meadow".

When groups of two pupils drew together, a Mac Book was added to the sketch session to record a video of the sketch creation process.

7.2 Results

The following observations were made during the app test:

- During pairing process one child usually understood the concept first and immediately told the others how to connect.
- Pupils usually started with drawing simultaneously. After a few seconds they recognized that this approach is not target-aimed. Then they began to discuss their drawing approach. Most of the groups divided the drawing work. One pupil drew the sky, sun and clouds, others drew flowers and animals. A group of two girls divided their drawing work strikingly: one girl drew the shape of the houses, the other girl colored the houses, they seemed to be very harmonic performing their work.
- Some pupils got very angry because of others who drew in their area or delete their shapes and some of them even refused to continue drawing.
- Others followed some pupils' commands blindly. They even asked others how to colorize their own shapes.
- The first two groups were told that they could start over again if they have troubles. Teamsketch has a button to clear the screen.
  This led to fast restarts whenever they encountered a discussion what and where to draw. All other groups did not get the information how to clear the sketch therefore they had to solve their issues to continue.
- All pupils were introduced to the zoom function but no one used this function although they frequently got the hint. They preferred to see the sketch as a whole.

[1] https://itunes.apple.com/at/app/buchstaben-post/id736836885?mt=8, last accessed: 17. Nov. 2014

Table 7.1: Test questionnaire - ranking questions

Question	Rating (average of all groups)
Did you like drawing together	2
Was the app easy to use	1
Would you like to play again	1

- Pupils did not understand how to switch the felt pens from broad to fine (double tap).
- Some pupils looked for a rubber. Others realized that the white color felt pen takes over this functionality.
- Pupils accidentally changed felt pen colors by touching the pens unintentionally.
- One child preferred to draw alone, he did not like drawing simultaneously.

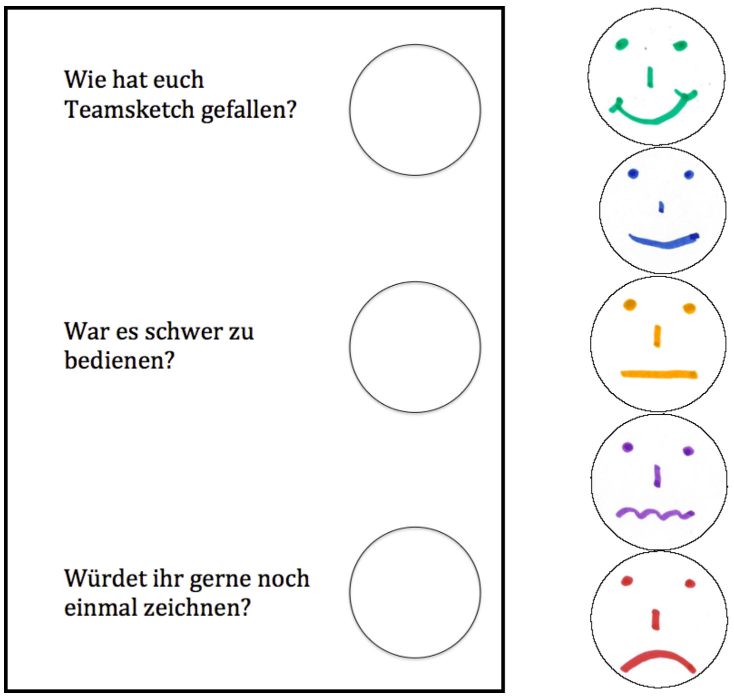
7.3 Questionnaire

After the app test the pupils were asked three rating questions and two open questions. The questions could be rated by selecting an appropriate smiley. Each group had to discuss the questions and rate the question as the whole group. There was no individual rating. Five smileys were used for rating. The happiest smiley corresponds to school mark ″1″, the saddest smiley corresponds to school mark ″5″. The main purpose of this questionnaire was not to get a rating, the idea was to get information about their impressions. While they were discussing the rating, they revealed their real opinion about the app. Table 7.1 elaborates on the question details.

Additionally two open questions were asked:

- Did you miss anything in this app?
  Answer: Some pupils would have liked to have importable shapes or images as well as sound effects. Others would have liked more different colors especially shiny colors such as gold.
- Would you have liked something to be different in this app?
  No answer was given to this question.

Figure 7.2 shows the questionnaire and rating equipment.
Most of the pupils liked drawing together. Many pupils wanted more shiny and glossy colors as well as importable custom shapes such as cliparts and photos. Some pupils missed sound effects.


Wie hat euch Teamsketch gefallen?

War es schwer zu bedienen?

Würdet ihr gerne noch einmal zeichnen?

Figure 7.2: Questionnaire and rating equipment

7.3.1 Impressions of the pupils given during the questionnaire

The following information was given by the pupils while they discussed the rating.

- sound effects
  Some pupils mentioned that they missed sound effects while drawing.

- drawing together
  While they discuss if they liked drawing together, some pupils and groups complained about other members who disturbed their painting process. These groups rated the first question significantly lower than other groups.

- additional features
  Some pupils mentioned that they would like to have more shapes and colors.

Usually individual pupils just decided the group's mark. The pupils had to be motivated to discuss the questions.

7.4 Teacher's feedback

The class teacher has four years experience in using iPads in class. Therefore, her feedback should be considered as well. She was very pleased that the app does not have any sound effects, the class atmosphere was more silent and comfortable without any additional noise. She also mentioned that the felt pen switching with double tap was not intuitive. The class teacher rated the collaborative learning effect as very high. She also mentioned that she will continue using the app in class to improve the team skills of their pupils.

8 Conclusion

The technical progress of tablet devices enables more and more use cases in educational environment. The device responsiveness in multi-peer networks increased a lot due to more computational power of the devices. With these new technologies time-critical applications could be implemented very efficiently which opens new possibilities for educational games. By joining additional observing devices, more information about the pupil's personal learning progress could be received.

8.1 Project Summary

The project consists of three main parts:

- web service
  The web service provides the API to upload and download images as well as user authentication.

- web interface
  A platform for teachers as well as pupils was implemented to access stored images. Teachers could use this web service to evaluate the pupils progress.

- app
  The iOS app Teamsketch was implemented as collaborative drawing application for two to four players. User draw simultaneously on one sketch to train collaborative learning and other team skills.

The implementation of the Teamsketch app addressed several issues of other already available apps. A curve fitting and line smoothing algorithm improve the line, therefore shapes and strokes look more realistic. The app uses peer-to-peer connections, therefore an internet connection is not necessary. Bluetooth connections can be used as well. The various connection capabilities are a big advantage while using the app in school environment. The app does not depend on a fast network infrastructure. The sketch synchronization between all devices is very responsive, no stroke synchronization errors happened during tests as well as the field study.
The web service and web interface provide a sketch store and retrieve platform which enables teacher to review student sketches easily.
The field study showed the potential of the app to train team and collaboration skills.

8.2 Technical Aspects

The development was started with Swift in beta state this increased the difficulty level significantly. Every time Apple updated the beta versions a lot of code parts stopped working and had to be changed. The public release of iOS 8.0 and 8.1 was shortly before Teamsketch development was finished. Therefore multiple issues with performance on older devices had to be solved. Debugging and testing with iOS simulator was misleading, because the host computer had much more computational power than the real devices. Therefore the simulator was not able to detect computation limitations. The app had big performance issues on the iPad 3 (first high resolution iPad). On this iPad the graphic performance is significantly lower than on the older iPad 2 model caused by the four times higher number of pixels[1]. Algorithm changes had to be implemented addressing this issue.

8.3 Lessons Learned

The following lessons had been learned while implementing and testing Teamsketch:

- tests with all devices
  Performance critical apps must be tested with all available types of devices. The issue with the iPad 3 came up in the last two weeks which aggravated the punctual release significantly.

- pairing mechanism optimization
  Most of the pupils were very impatient while pairing. In this situation other skills such as patience and consideration could be trained. They learn that they have to wait until all devices are listed and that it takes time to set up the environment.

- double tap to change felt pen width was not intuitive

- pen color changes happened accidentally
  A better pen change procedure should be considered

- undo function
  Pupils should not have the possibility to easily undo (such as the common undo function) or reset their work because for some pupils this was the easiest solution to escape discussions and issues.

- zoom function may not be necessary
  Most pupils prefer to see the whole sketch.

[1] http://support.apple.com/kb/sp647, last accessed: 15. Nov. 2014

- collaboration and team skills
  These skills should be trained as early as possible as many pupils had issues with working in teams.

8.4 Further Studies

Teamsketch field test showed that drawing together simultaneously needs high developed collaboration skills. These skills could be trained while using the app. By only observing the pupils for a short time, the social role of each pupil was revealed. Further studies should be made with experienced observers from the field social learning, psychology and education to evaluate collaborative learning with Teamsketch.

Appendix

Bibliography

Callahan, Brooke and Wil Turner (2014). "Testing in Xcode 6." In: *Apple Worldwide Developers Conference 2014*. 414. Apple Inc. 1 Infinite Loop Cupertino, CA 95014, USA: Apple Inc., p. 180 (cit. on p. 57).

Chiu, K., M. Govindaraju, and R. Bramley (2002). "Investigating the limits of SOAP performance for scientific computing." In: *High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings. 11th IEEE International Symposium on*, pp. 246–254. DOI: 10.1109/HPDC.2002.1029924 (cit. on p. 61).

Christensen, Erik et al. (2001). *Web Services Description Language (WSDL) 1.1*. URL: http://www.w3.org/TR/wsdl (visited on 11/15/2014) (cit. on p. 64).

Danesh, Arman et al. (2001). "GeneyTM: Designing a Collaborative Activity for the palmTM Handheld Computer." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '01. Seattle, Washington, USA: ACM, pp. 388–395. ISBN: 1-58113-327-8. DOI: 10.1145/365024.365303. URL: http://doi.acm.org/10.1145/365024.365303 (cit. on pp. 4, 5).

Dillenbourg, Pierre (1999). *Collaborative learning: cognitive and computational approaches*. 2nd Revised Edition. Emerald Group Publishing Limited (cit. on p. 3).

Farin, Gerald (2002). *Curves and Surfaces for CAGD: A Practical Guide*. 5th. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN: 1-55860-737-4 (cit. on pp. 39, 40, 54).

Gamma, Erich et al. (1995). *Design Patterns. Elements of Reusable Object-Oriented Software*. 32nd Printing. Addison-Wesley (cit. on pp. 21, 22).

Grimus, Margarete and Martin Ebner (2014). "Learning with Mobile Devices Perceptions of Students and Teachers at Lower Secondary Schools in Austria." In: *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2014*, pp. 1600–1609 (cit. on p. 1).

Hongping, Shu and Wei Zhaoyu (2012). "Contour Smoothing Algorithm Based on Bezier Curves and Application." In: *Computational Intelligence and Communication Networks (CICN), 2012 Fourth International Conference on*, pp. 283–285. DOI: 10.1109/CICN.2012.73 (cit. on p. 54).

Isted, Tim and Dave Addey (2014). "Introduction to Swift." In: *Apple Worldwide Developers Conference 2014*. 402. Apple Inc. 1 Infinite Loop Cupertino, CA 95014, USA: Apple Inc., p. 343 (cit. on p. 15).

Jobs, Steve (2010). *Thoughts on Flash*. Ed. by Apple Inc. URL: https://www.apple.com/hotnews/thoughts-on-flash/ (cit. on p. 11).

Johnson, David W. (2003). "Social Interdependence: Interrelationships Among Theory, Research, and Practice." In: *American Psychologist* 40.11, pp. 934–945. ISSN: 0003-066X (cit. on p. 4).

Johnson, David W. and Roger T. Johnson (1989). *Cooperation and Competition: Theory and Research*. Interaction Book Co (cit. on p. 3).

Johnson, David W. and Roger T. Johnson (2005). "New Developments in Social Interdependence Theory." In: *Genetic, Social, and General Psychology Monographs*, pp. 285–358. ISSN: 8756-7547 (cit. on p. 4).

Johnson, David W. and Roger T. Johnson (2008). "Wie kooperatives Lernen funktioniert." In: *Friedrich Jahresheft* 26, pp. 16–20. ISSN: 0176-2966 (cit. on p. 4).

Johnson, David W., Roger T. Johnson, and Edythe Johnson Holubec (2002). *Circles of Learning: Cooperation in the Classroom*. 5th Printing. Interaction Book Company (cit. on pp. 3, 4).

Kienleitner, Benedikt (2013). *A Contribution to Collaborative Learning Using iPads for School Children* (cit. on p. 84).

MacIntyre, Peter, Brian Danchilla, and Mladen Gogala (2011). *Pro PHP Programming*. Apress (cit. on p. 57).

Melzer, Ingo (2010). *Service-orientierte Architekturen mit Web Services*. 4th Printing. Spektrum Akademischer Verlag Heidelberg (cit. on p. 63).

Meszaros, Gerard (2007). *XUnit test patterns : refactoring test code*. 1st Printing. Pearson Education, Inc. (cit. on pp. 57, 58, 82).

Mhatre, H.K., B.A. Mehta, and A.K. Jaiswal (2013). "Architecture for MTOM based file transfer." In: *Circuits, Power and Computing Technologies (ICCPCT), 2013 International Conference on*, pp. 1250–1252. DOI: 10.1109/ICCPCT.2013.6529002 (cit. on p. 67).

Papazoglou, M.P. et al. (2007). "Service-Oriented Computing: State of the Art and Research Challenges." In: *Computer* 40.11, pp. 38–45. ISSN: 0018-9162. DOI: 10.1109/MC.2007.400 (cit. on p. 61).

Rideout, Victoria (2013). *Zero to Eight*. Tech. rep. Common Sense Media Inc. (cit. on pp. 1, 5).

Zurita, Gustavo and Miguel Nussbaum (2004). "Computer supported collaborative learning using wirelessly interconnected handheld computers." In: *Computers and Education* 42.3, pp. 289–314. ISSN: 0360-1315. DOI: http://dx.doi.org/10.1016/j.compedu.2003.08.005. URL: http://www.sciencedirect.com/science/article/pii/S0360131503000927 (cit. on p. 4).