Honours Project Report

eSports Portal Mobile Application

By Hayoung Noh

Supervised by:
Associate Professor Sonia Berman

	Category	Min	Max	Chosen
1	Requirement Analysis and Design	0	20	15
2	Theoretical Analysis	0	25	0
3	Experiment Design and Execution	0	20	0
4	System Development and Implementation	0	15	15
5	Results, Findings and Conclusion	10	20	15
6	Aim Formulation and Background Work	10	15	15
7	Quality of Report Writing and Presentation	10		10
8	Adherence to Project Proposal and Quality of Deliverables	10		10
9	Overall General Project Evaluation	0	10	0
Total Marks		**80**		**80**

Department of Computer Science
University of Cape Town
2014

Abstract

Derivco, a game development company, initiated a website project to facilitate in-house Electronic Sports Leagues for the employees. In this project, a mobile application is needed to take a photograph of the match result screen and upload this photograph and the final score to the server. This application then further provides functionality that a user is likely to need on a mobile, such as checking for upcoming matches and recent results. This project covers the development of the eSports mobile application as well as the web-service for the mobile application to facilitate data exchange between the server and the mobile application.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Problem Outline

Derivco is a global software development company that has recently acquired various other companies around the globe. Each of Derivco’s offices contains a gaming console which is used to play games competitively with employees in other offices. This is done to build friendly relations between the members of different offices, particularly those newly acquired from previously competing companies. To this end Derivco is setting up eSports leagues to manage competitions. They wish to manage the leagues via a web-based system which would allow for the managing of players, teams, leagues, results, game fixture calendars, etc. In order for a team to upload their match results to the website, a mobile application is needed to capture the results and upload to the website server. The results include a screenshot of the match result and the score for the match. Other player functionality that is likely to be used on a mobile phone, such as checking for upcoming matches, should also be provided.

1.2 Work Division

A team of three members was assigned for the eSports Project. The work was split into three areas: player website, admin website and mobile application. Player and admin website is a single website with player-side and admin-side handled separately. The work was split in this manner because Derivco strongly preferred this to having each team member develop a separate layer of the system.

Initially, the web-service was assigned as part of the website development, but during the project it had to be re-assigned to the mobile development due to the web-service being left out in the website architecture. This report covers the mobile development side of the project which includes the Android application and the web-service.

1.3 Proposed System

The aim of the project is to develop a mobile application that has the main functionality of capturing and uploading the results as well as a web-service that handles the communication between the mobile application and the database of the server. Additional functionality could then be added according to what was most useful to do on a mobile phone.

1.4 Report Outline

This report is divided into several sections that include: background chapter which reviews general mobile application development, requirement analysis that covers user requirements for the mobile application, design and implementation chapters that cover mobile development and web-service development, user testing and evaluation, future work and conclusion which includes a discussion of certain problems encountered in the project.

2. Background Chapter

When starting a mobile development, few questions may be raised. What differentiates mobile application development and traditional application development? What options are available? What are the important design principles for mobile application? What is a web-service? These questions are discussed in this chapter.

2.1 Software Engineering Issues in Mobile Development

Software engineering for mobile application shares similar practices with traditional application, but some specific issues need to be addressed in mobile development.

The first point is the potential interaction of applications with each other [26]. Mobile devices have many applications from various sources with the possibility of interaction between them. Secondly, the accelerometers that respond to device movements, numerous touch screen gestures, global positioning system, microphones that are usable in applications other than voice calls, cameras, and multiple networking protocols are all in a single device, allowing many feature options for the application.

Mobile applications are required to support multiple devices with various screen sizes and different hardware. In addition, different versions of the operation systems are released much more frequently than the embedded devices complicating the support [26].

Lastly, many aspects of an application may affect the device's power, draining the battery life of the device.

The following issues were raised in [26]:

- Determining which functions should be present in a mobile version of a traditional application.
- Providing techniques that can assure maximum reuse of code among different versions.
- Determining comparable effort in building a native mobile application to a mobile web application.
- Determining if the mobile user interface requires a different contextual design process to support a different set of use cases.
- Integrating the various forms of input and sensor data in application design.
- Determining if the synchronization techniques from traditional client-server computing would suffice in the potential loss of connectivity or battery power running out.
- Different application design depending on the speed of the network.
- Creating an application that can maximize battery life and resource usage.

The best practice suggested by [26] was to follow an agile methodology that can quickly adapt to changing user requirements and to follow development guidelines published by various platforms.

2.2 Web vs. Native Application

Before going into different mobile operating systems, it would be worth considering mobile web application versus native application. Depending on the type of application, mobile web application could simplify the development in terms of time and cost.

It was argued in [11] that the performance of native applications will only be noticed for high image processing or 3D games. It was also pointed out that the native application languages are generally known to be more complicated then Web application languages.
One of the disadvantages for Web applications was that the standard APIs for Web application interfaces are much weaker than the native applications. The scaling of Web interfaces on different platforms and devices was also raised as an issue [11].

User experience is another area to have an effect on both native and web application development. Users may have a different expectation for a native application than for the web application. Web applications must be connected to the Internet the entire time the application is running but native applications can work offline as well as online.

In the area of performance, the size of the payload and the interpreting of code influence how fast the web application can run. The conclusion reached in [11] was to favour data over decoration in the web application. The advantage of a native application is that performance issues are not related to the size of payload and the code is already compiled.

2.3 iOS and Android Operating System

Today, there are at least five important platforms (iPhone, Android, Blackberry, Windows Phone, Symbian) [26], but detailed examination of all the platforms will be impractical. Also, a large portion of the mobile markets in the world are currently iPhone and Android [22]. Thus, only the two platforms were reviewed.

Comparisons of the two platforms in [15] are summarized in Table 1:

Table 1: Comparison Between iOS and Android

	iOS	**Android**
Minimum Operating System Requirement for Development	• Macintosh computers running Mac OS X 10.6 (Snow Leopard)	•Windows XP •Linux •Mac OS X 10.5.8
Language	•Objective-C	•Java (Dalvik VM) •Scripting (SL4A) •LogoBlocks (AppInventor)

IDE	•Xcode	•Eclipse 3.5 (requires Android SDK plug-in) •Can sign applications for release in the Android Market
Garbage Collection	•Not available for performance reasons	•Available
GUI Creation	•Xcode	•XML
Simulator	•iOS SDK bundled •GPS and Accelerometer not supported	•Android Virtual Devices (AVDs) that run on the Android Emulator •GPS and phone/SMS interrupt signals that can be passed to the emulator through telnet connection •Accelerometer, orientation and compass readings can be manipulated using OpenIntent's SensorSimulator
Graphics	•OpenGL (support for 2D and 3D)	•OpenGL (support for 2D and 3D)
Database	•SQLite	•SQLite
Tutorial Resources	•Apple's technical documentation provides vast information	•Official tutorial development guide is available
Reference Website	http://developer.apple.com/iphone	http://developer.android.com

2.4 Human-Computer Interaction in Mobile

User interface is an important part of any software application that has user interaction. Even if the industrial design and their aesthetic are appealing, if it does not address real user needs, it will be of no use to the user [20].

Constant testing and refinement of the design by engaging with the actual users are vital. Various methods and activities can be used to find out what matters for the users. The most effective way to get user feedback on the design is through a prototype [20]. Different forms of prototypes range from low-fidelity paper-based sketches to complex pieces of software written for specialist hardware.

There were a few design guidelines that stood out in [20]. The limiting of the number of items in the menu to a maximum of seven was one of them. Use of icons (symbols) was suggested to replace text whenever possible to free-up space. Generally, pictures are easier to remember than text. The authors argued that well-designed overviews are powerful features that can overcome restrictions of small screen sizes. Overviews are a zoomed-out version of the displayed contents. By having an overview, the users can easily orientate themselves relative to the overall content. Other design guidelines included quick navigation to frequent functions, limiting excessive scrolling or page-to-page navigation and stroking not poking [20]. Excessive scrolling is tiring on a small screen and users can get lost in a page-to-page navigation. Stroking (swiping) interaction gives more flow and faster navigation then poking (pressing). Stroking is also less tiring then poking.

2.5 Web-service for Mobile Application

A web-service is needed in order to upload and retrieve data to the database from the mobile application. Web-services are interfaces and bindings that can be defined to support direct interaction with other software by exchanging messages via internet-based protocols [1]. RESTful web-services were reviewed for this project due to being less intensive in memory and processor compared to the traditional web-services.

Representational state transfer (REST) is an architectural style that only allows create, read, update and delete (CRUD) operations to the database [21]. Traditional web-services are memory and processor intensive to the client. For the limited memory and processing power of the mobile devices, RESTful web-services are easy to invoke, can produce a discretely formatted response that can be easily parsed, and has no TCP connections to slow down the message exchange [12].

Key aspects of REST were given in the following summary:

- REST is stateless
  - Impact of network volatility is minimized
- REST is URL based
  - This makes it easy to invoke
- REST responses are HTTP based
  - Making it discrete
  - Also minimizes the impact of network volatility
- REST delivery can be made very concise and compact
  - No excessive protocol elements
  - Memory usage for client can be minimized

2.6 Conclusion

Five areas that can be helpful in starting the eSports mobile development were presented. Some issues in the software engineering are without solutions. These issues should be considered for complex applications and need further research. However, for the eSports mobile application, iterative software engineering will suffice. In deciding whether to develop a native application or web application, a native application was chosen due to the supportive libraries from the native application platforms and no need for a constant network connection to use the application.

For native application platforms, Android and iOS were compared. Both platforms have advantages but due to the free development environment of Android and having an Android device available to test the application, Android was chosen.

The HCI section provided insight in designing user-centered applications. User requirements and functionality will be the focus in the eSports application design.

In the last section, web-services were introduced. Specifically, how the RESTful services are best suited for the mobile application due to the low memory and processor required in exchanging data between the web-service and the mobile application were discussed. The eSports web-service will implement the RESTful architectural style.

3. Requirements Analysis

3.1 Functionality Requirements

Derivco requested the following requirements for the eSports project:

- Manage player registrations (sign up).
- Allow players to create/join teams.
- Allow administrators to create leagues with progression information (i.e. round robin, knockout ladder or both).
- Administrators must be able to create and manage league rules.
- Generate fixtures based on availability windows.
- Submit results (win/lose or win/lose/draw) and scores.
- The system should accommodate uploading of screenshots/photos as proof of the result.
- Dispute management process for league administrators (in the case Team B disputes the results submitted by Team A).
- Results and fixtures page.
- News page where administrators can create and manage news articles.
- Game calendar – players should be able to view a calendar showing their upcoming fixtures.

The following requirements were assigned to the eSports mobile application:

- Take photograph of the match result screen.
- Upload photo to the server.
- Upload match score and result to the server.
- View uploaded results.
- View upcoming fixtures.
- View leader-board.
- View team ladder – fixtures displayed in a ladder (for knock-out stages).
- View/comment dispute process – show uploaded result photo and have comment thread under the photo.

Web-service requirements include:

- Use Representational state transfer (REST) architectural style with the GET and POST operations (REST is less intensive in memory and processing for the client).
- Verify user credentials.
- Retrieve/save data from eSports database and send/receive data to/from mobile application.

Due to the re-assigning of the web-service, the last three requirements of the mobile application, view leader-board, view team ladder, and view/comment dispute process, were left out from the mobile development.

3.2 System Requirements

Derivco asked to have a completely separate system for the mobile application. The only shared resource with the web development team is the central database. However, the web-service was integrated with the website to access the central database.

Derivco did not specify any mobile platform they preferred. When the question was raised concerning the platform, they listed Android, iOS or Web application and gave the freedom to choose which platform to develop. Android was chosen over iOS as the mobile platform due to the free development environment. Web application was discarded due to the nature of the Web application constantly requiring the online connection. Also, the native application platform offered better support libraries.

3.3 Requirements Recording

After the analysis of the requirements, use case diagram was made to depict the core features of the mobile application and the web-service.

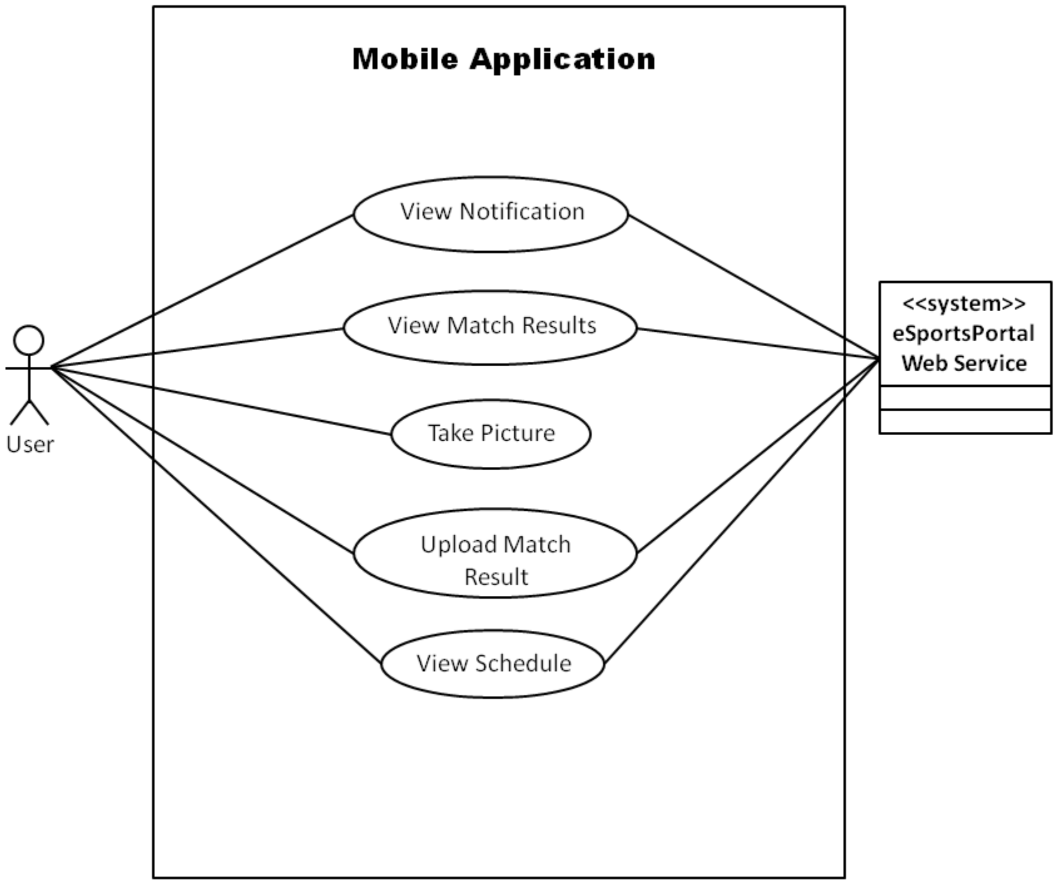


Figure 1: Mobile Application Use Case Diagram

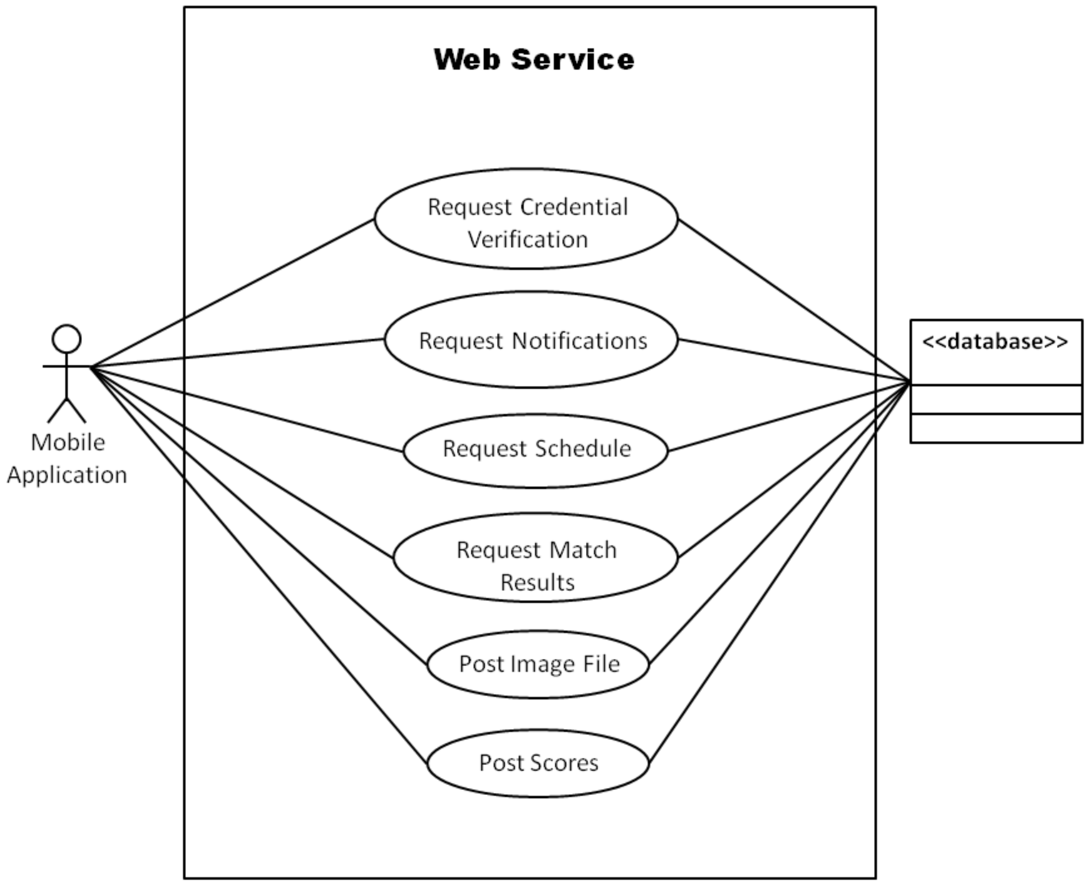


Figure 2: Web-service Use Case Diagram

4. System Overview and Interface Design

4.1 System Architecture

This section presents the overview of the Android application and web-service architecture layers and the package view of the communication flow between them.

4.1.1 Package Overview

Android application uses Java AsyncTask which creates a background thread to connect to the web-service and requests/sends data. AndroidRestService processes the request and queries/updates the database and returns data/response. AsyncTask handles the returned data/response and updates the main user interface (UI) thread of the application.

The flow of communication between the Android application and the web-service is shown in Figure 3.

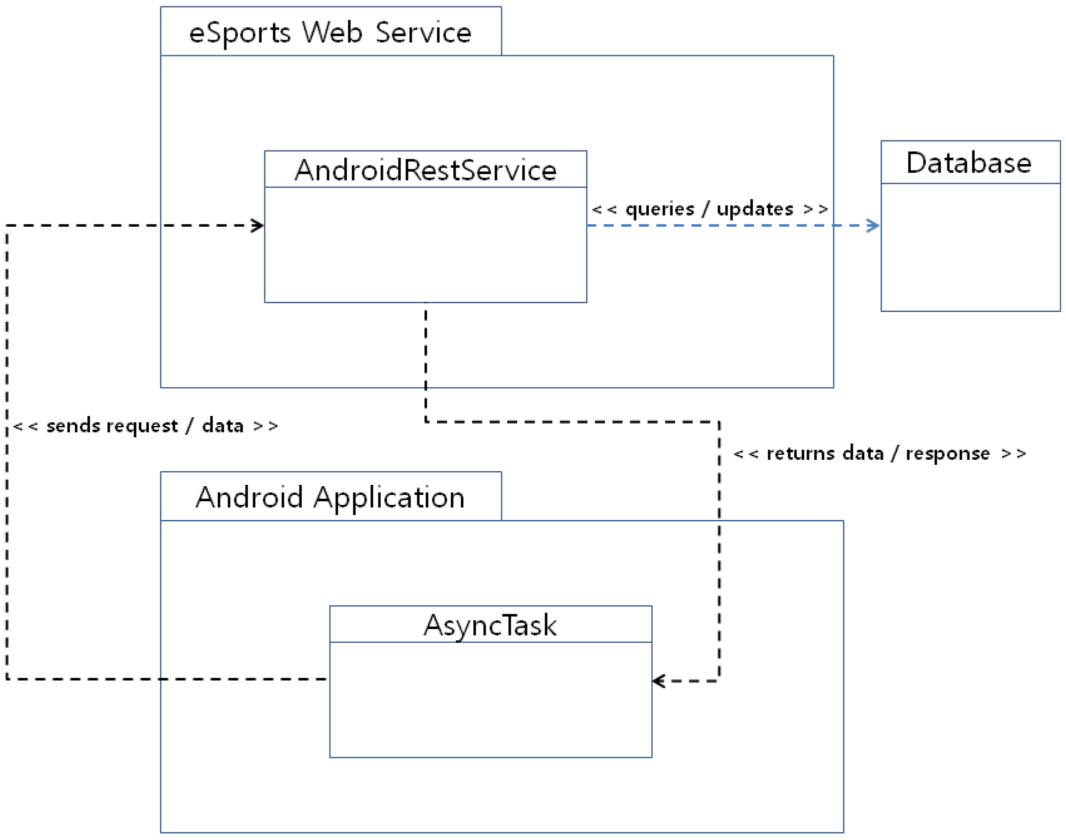


Figure 3: Package Diagram

4.1.2 Android Architecture

eSports application uses the android framework with the supporting libraries. HttpClient and Joda Time libraries are external open source libraries. HttpClient library offers the Multipart Entity framework for sending multiple data and Joda Time library offers calculation of the difference between two dates. The use of the two libraries will be discussed in more detail in the following implementation chapter. At run time, Dalvik Virtual Machine pulls all the libraries with the compiled source code and executes the application.

Figure 4 shows the Android architecture layers.

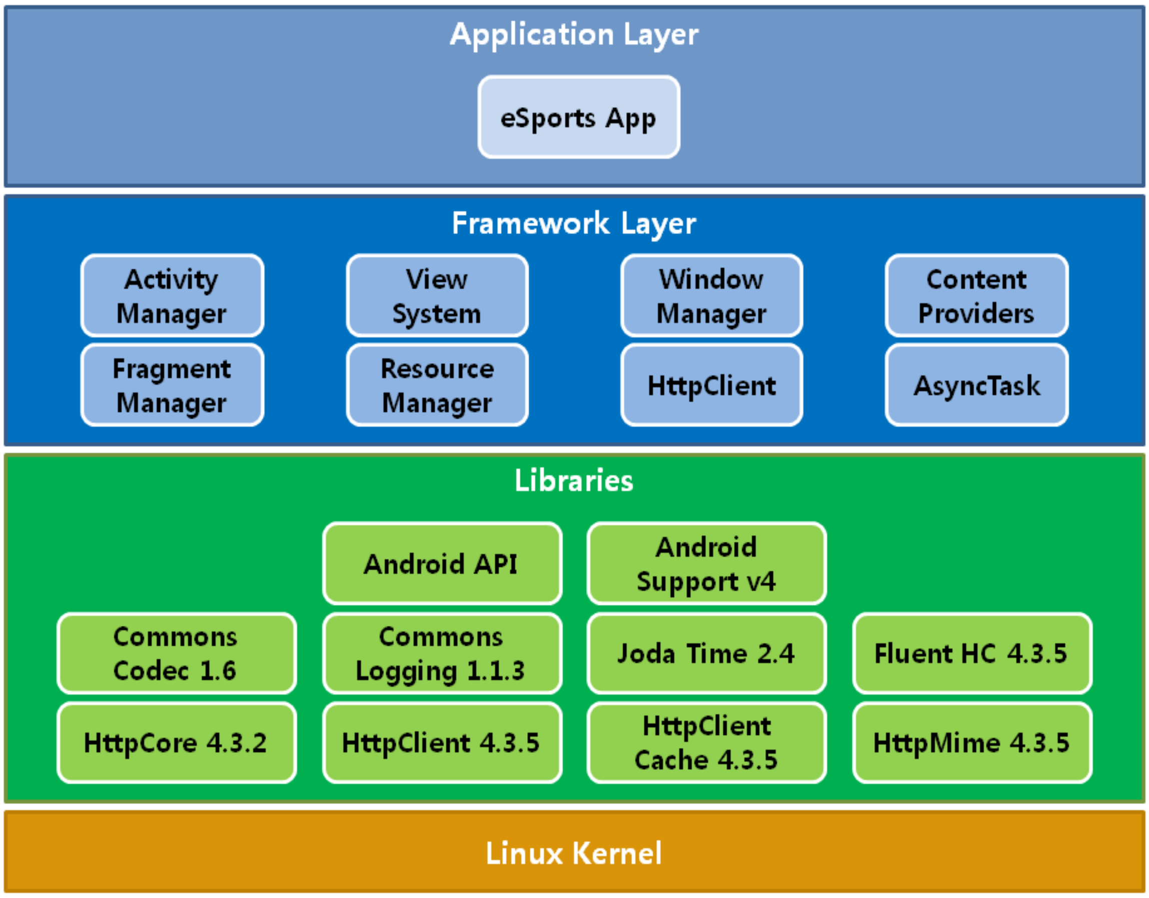


Figure 4: Android Application Architecture

Figure 4: Adapted from TutorialsPoint
http://www.tutorialspoint.com/android/android_architecture.htm
[Accessed 28-10-2014]

4.1.3 Web-service Architecture

Web-service uses the URL Access Layer as the gateway for sending and receiving data from the eSports application.

Requesting Data:

A Data request is sent to the data query layer to formulate a SQL query command for retrieving data. The query layer opens the connection to the database with the connection string. The query command is sent to the database. Queried data is then returned to the URL Access Layer.

Uploading Data:

Uploaded data is sent to the query layer to formulate a SQL query command for updating or adding data. The query layer opens connection to the database with the connection string and checks whether to update or add new data. Once the updating or adding is determined, the command string is sent to the database through a new connection. The Result of the database update is then returned to the URL Access Layer.

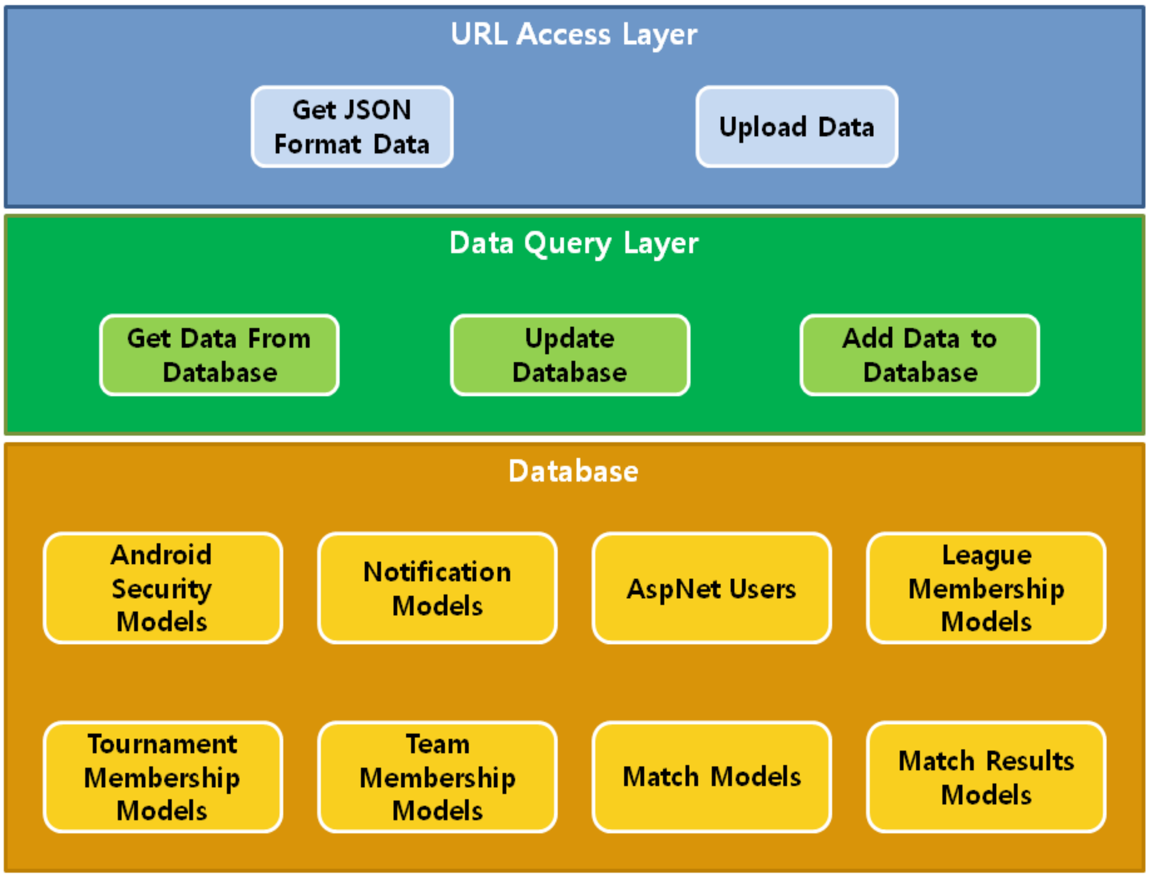
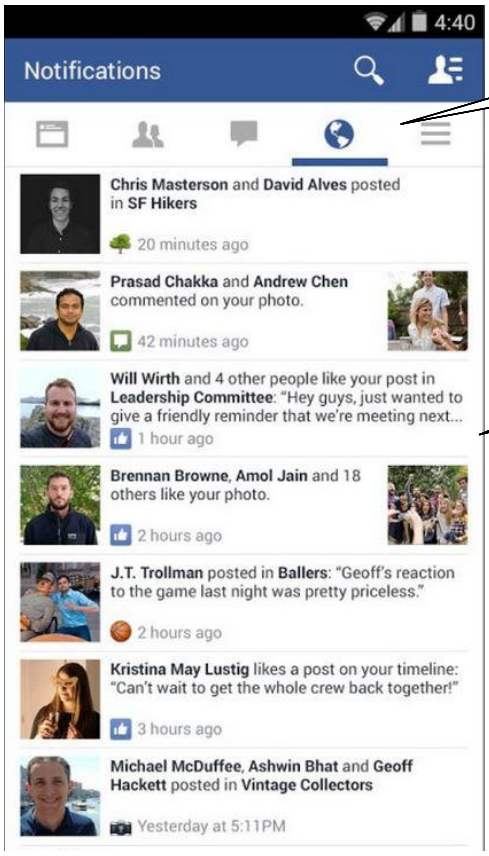


Figure 5: Web-service Architecture

4.2 Interface Design Inspiration

The design inspiration for the eSports application came from Facebook and Kakaotalk. Facebook is a well-known social media application and Kakaotalk is widely used in Eastern Asia as the messenger application equivalent to WhatsApp messenger application. Both Facebook and Kakaotalk use the tab menu option with the contents shown in a scroll view. These apps also allow for side swiping to switch to different tab options. The reason for the adaptation of these design features is the similarity of information that needs to be shown to the user between eSports application and these social media applications.

eSports application's core information that needs to be shown are: list of past matches that were played; list of match results and notifications; list of upcoming matches. These lists of information are usually longer than the full screen page and the scroll view can facilitate this well. The android design guide highly recommends the tab menu options when the options are limited to three or four menus [9]. In addition to the tab menus the swiping functionality to switch between different menus allows for less poking and more stroking principle from Jones and Marsden (2006).



Chris Masterson and David Alves posted in SF Hikers
20 minutes ago

Prasad Chakka and Andrew Chen commented on your photo.
42 minutes ago

Will Wirth and 4 other people like your post in Leadership Committee: "Hey guys, just wanted to give a friendly reminder that we're meeting next...
1 hour ago

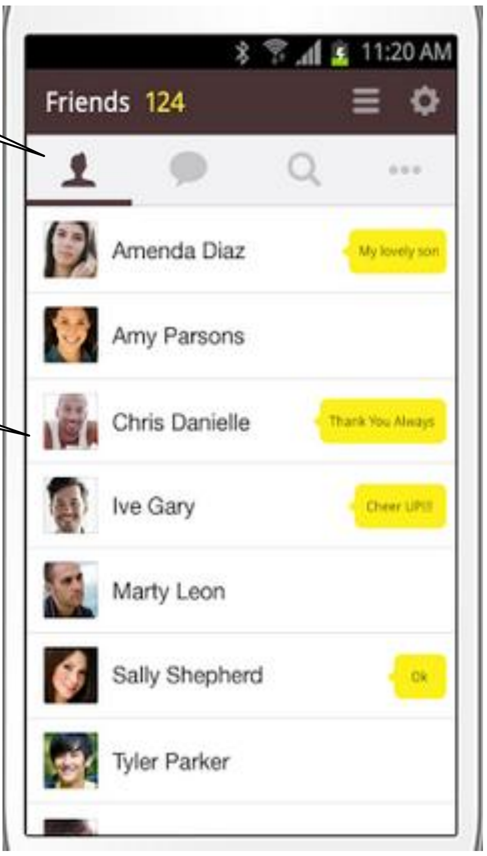
Brennan Browne, Amol Jain and 18 others like your photo.
2 hours ago

J.T. Trollman posted in Ballers: "Geoff's reaction to the game last night was pretty priceless."
2 hours ago

Kristina May Lustig likes a post on your timeline: "Can't wait to get the whole crew back together!"
3 hours ago

Michael McDuffee, Ashwin Bhat and Geoff Hackett posted in Vintage Collectors
Yesterday at 5:11PM

Figure 6: Facebook Notifications Screen



Amenda Diaz — My lovely son
Amy Parsons
Chris Danielle — Thank You Always
Ive Gary — Cheer UP!!
Marty Leon
Sally Shepherd — Ok
Tyler Parker

Figure 7: Kakaotalk Friends Screen

Figure 6: http://androidever.com/facebook-download/ [Accessed 28-10-2014]
Figure 7: http://www.kakaotalkdownload.com/download-kakaotalk-v4-3-6-for-android [Accessed 20-10-2014]

4.3 Paper Prototype

Paper prototype was made to test the tab menu interface and the general interaction of the application.

4.3.1 Initial Design

The main screen views shown in Figures 9-12 were drawn following the design inspiration. From left to right the tab menus are Upload, Home, Schedule and Results. After the login, the users will be taken to the Home screen which is the second tab option. This is to give the latest match results and the notifications first and then have the option to swipe left to upload a photo and swipe right to view the schedule. The design and navigation idea is to provide the users with easy access to the two most important features with a single swipe.

The image view in the Upload screen holds the logo of Derivco as a default image. When the photo is taken using the camera function, the captured photo will replace the Derivco logo and be shown in the image view. Schedule screen shows the upcoming matches. When the schedule box is pressed, the match statistic of the opponent team is shown as a pop-up. The Result screen shows the option to navigate to three different screens. These screens are Leaderboard screen, Ladder screen and the Dispute screen.

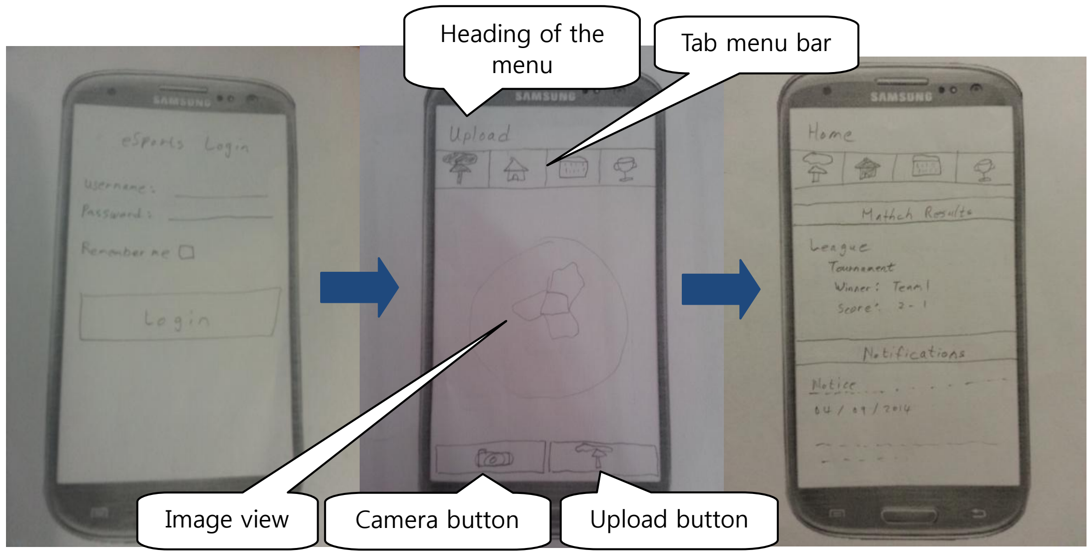


Figure 8: Paper Prototype Login *Figure 9: Paper Prototype Upload* *Figure 10: Paper Prototype Home*

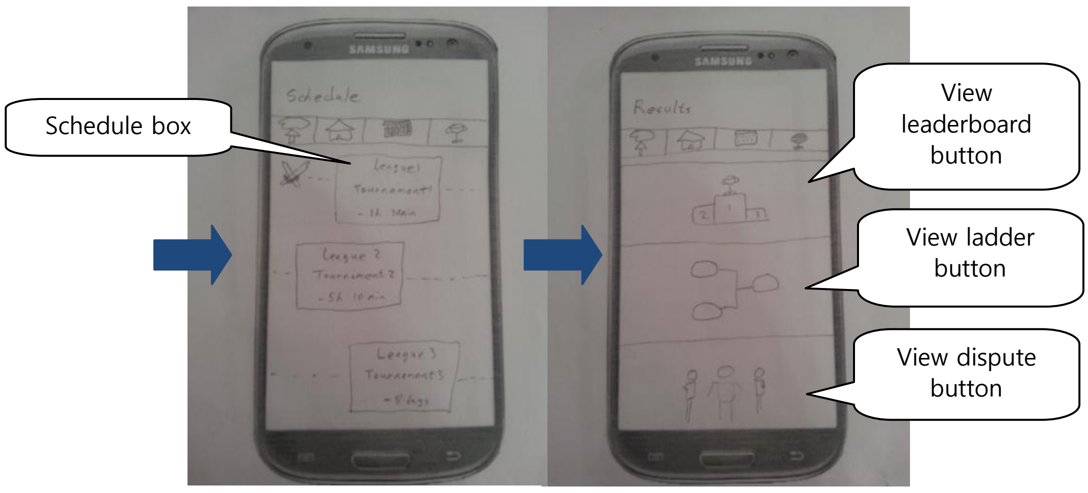


Figure 11: Paper Prototype Schedule *Figure 12: Paper Prototype Results*

4.3.2 User Testing

Two users, a first year Computer Science student and a software tester, volunteered to participate in the paper prototype user testing. Each user participated in the testing separately. Both users gave positive feedback on using the tab menu layout. However, they both showed concerns with the schedule icon. They could not recognize it as a calendar. They also did not know what the Derivco icon was, whether it was a button or not. The schedule box was missed by both of the users as well. They did not know that it was clickable. The swiping of the screen to navigate to the next menu was missed by one of the users.

After the user testing, it was concluded that a high-fidelity prototype is needed to further test if the icons would be more recognizable and the distinction between background image and buttons would be made clearer.

4.3.3 Design Change

Before implementing high-fidelity prototype, the upload screen had to be changed to show the list of past matches to choose from. The need for the change was brought up by the project team member. The reasoning was to cater for the multiple games that may have been played in a day and the user may need to upload scores for multiple matches. The design changed to have one camera button at the bottom and upload buttons on the side for each match. Each match is selectable and the photo taken using the camera button will be linked with the selected match and the photo will be shown in the image view. The first match on the list is auto selected by default. The selection will be shown by highlighting the selected section.

In addition to the design change, the time limit for the uploading of the results was discussed with the team members. In order for the tournaments to continue, the match results need to be uploaded (especially in the case of a knock out round). A 24-hour time limit was set to ensure that all the match results were uploaded within a day and it would be enough time for the users to upload the match results.

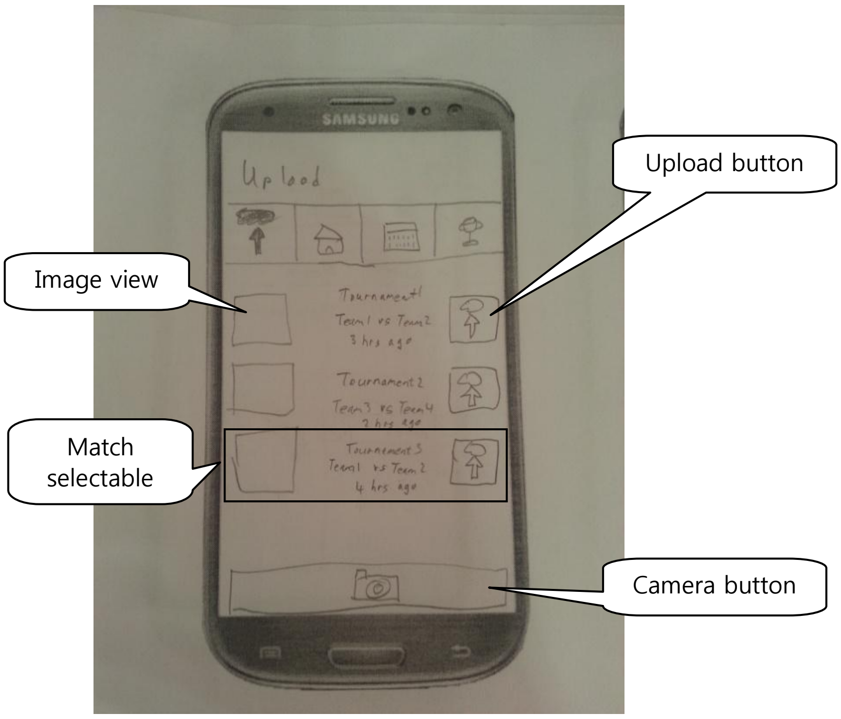


Figure 13: Paper Prototype Improved Upload

4.4 Icon Design

4.4.1 Icon Table

Various icons were made and for each icon four different sizes were made. This was to account for different screen resolutions: MDPI, HDPI, XHDPI and XXHDPI. The icon sizes were set according to the Android Guidelines from the Android Developer's website [6].

Two different colour versions of the icons were produced for different types of themes:

Table 2: Holo Light and Holo Dark Icons

	Holo Light Theme	**Holo Dark Theme**
Derivco Logo Icon		
Derivco Launcher Icon		
Camera Icon	📷	📷
Home Icon	🏠	🏠
Upload Icon	⬆	⬆
Schedule Icon	📅	📅
Results Icon	🏆	🏆
Battle Icon	⚔ – –	⚔ – –
Dash Short	– –	– –
Dash Long	– – – – –	– – – – –
Leaderboard Button	2 1 3	2 1 3

Ladder Button	⊢	⊢
Dispute Button	👥	👥
Screenshot Icon	🖼	🖼
Battle Statistic Icon	📈	📈
Win Icon	✔	✔
Loss Icon	💀	💀
Draw Icon	➖	➖
Dispute Icon	❗	❗
Refresh Icon	🔄	🔄
Notification Icon	✉	✉
Overflow Icon	⋮	⋮

4.4.2 Icon Design Feedback

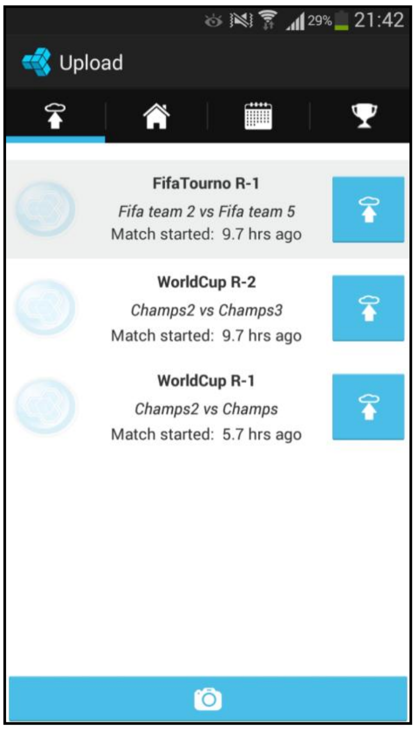
All the icons except the Derivco logo and Derivco launcher were made from scratch using Paint.Net for originality. However, the ideas for the design came from various sources found from Google search. The refresh icon, notification icon and the overflow icon were made from Android standard icons.

The design choices were made with three voluntary users. Two of the three users were the same users who participated in the paper prototype testing. One additional user works as an IT support for a mobile application company. The users participated in the design process one by one.

The process of designing icons involved: picking several possible images online, having the users pick the most appropriate image, creating the icon, discussing the color choice with the users. This process was repeated for all the icons. During the process there was a debate with one user about the upload icon due to the cloud image which indicates that the uploaded data is going to a cloud system. This was not resolved so the original design was kept. The rest of the icons had positive feedback. All the icons were clear enough for the users to understand what the icons meant given the context of the application. Not all icons were used in the final design due to some features being taken out in the process of adjusting the scope of the project.

4.5 Resulting Design

Figures 14 - 17 below show the 4 main screens that were created prior to the re-assigning of the web-service.

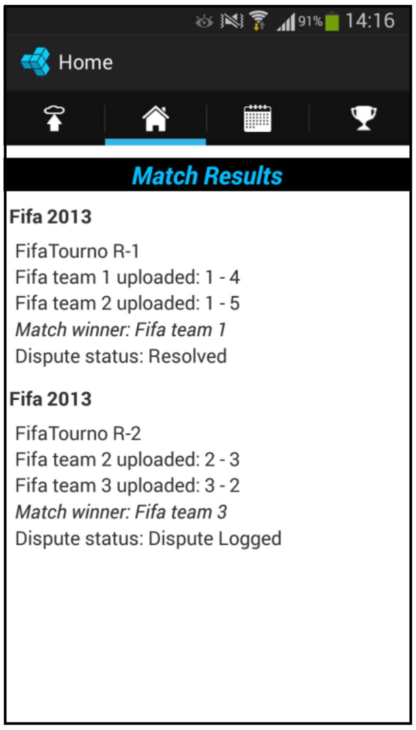

Upload

FifaTourno R-1
Fifa team 2 vs Fifa team 5
Match started: 9.7 hrs ago

WorldCup R-2
Champs2 vs Champs3
Match started: 9.7 hrs ago

WorldCup R-1
Champs2 vs Champs
Match started: 5.7 hrs ago

Figure 14: Upload Screen Before Scope Change

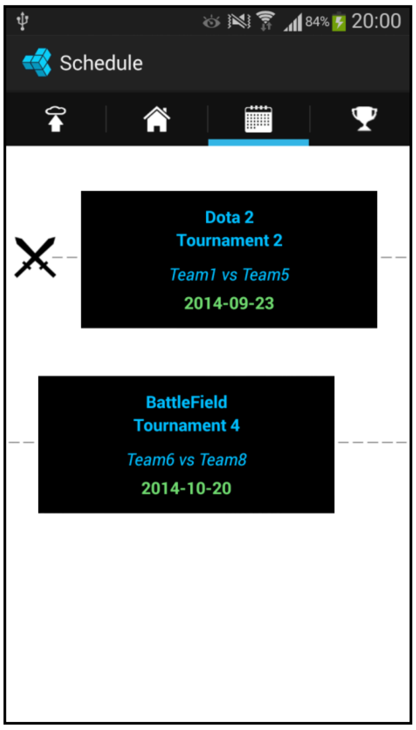

Home

Match Results

Fifa 2013
FifaTourno R-1
Fifa team 1 uploaded: 1 - 4
Fifa team 2 uploaded: 1 - 5
Match winner: Fifa team 1
Dispute status: Resolved

Fifa 2013
FifaTourno R-2
Fifa team 2 uploaded: 2 - 3
Fifa team 3 uploaded: 3 - 2
Match winner: Fifa team 3
Dispute status: Dispute Logged

Figure 15: Home Screen Before Scope Change


Schedule

Dota 2
Tournament 2
Team1 vs Team5
2014-09-23

BattleField
Tournament 4
Team6 vs Team8
2014-10-20

Figure 16: Schedule Screen Before Scope Change

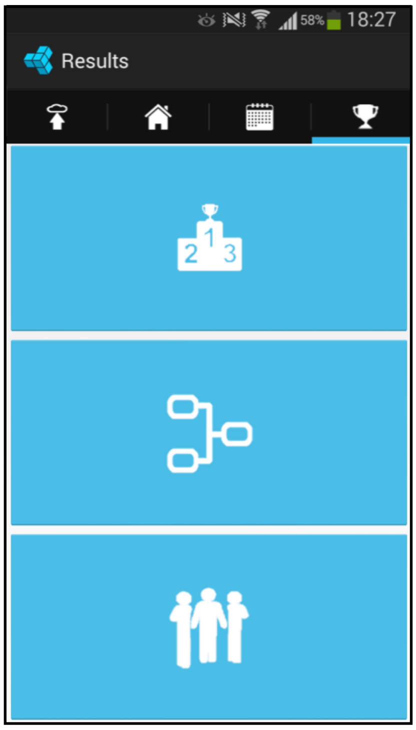
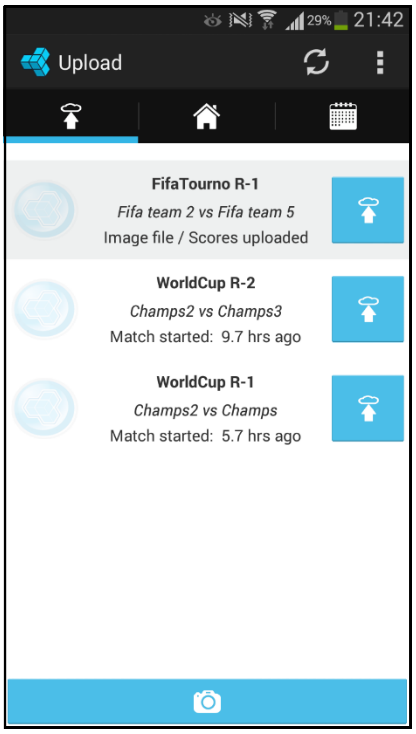


Figure 17: Results Screen Before Scope Change

Figures 18 - 20 show the 3 main screens that were created after the re-assigning of the web-service. The Results screen was removed and the colour of the schedule screen was changed. The next chapter includes details of how these were implemented.

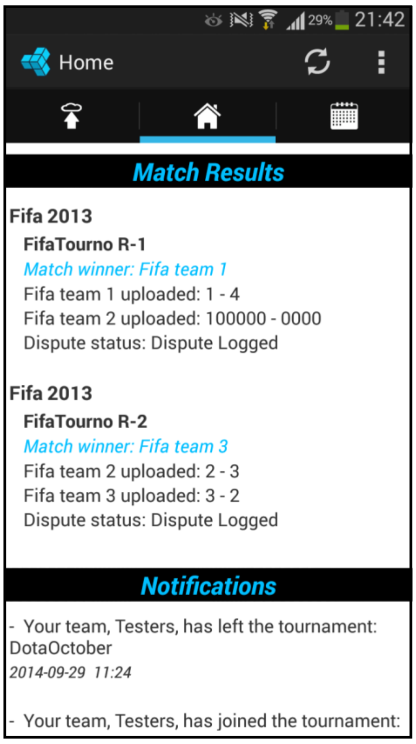


FifaTourno R-1
Fifa team 2 vs Fifa team 5
Image file / Scores uploaded

WorldCup R-2
Champs2 vs Champs3
Match started: 9.7 hrs ago

WorldCup R-1
Champs2 vs Champs
Match started: 5.7 hrs ago

Figure 18: Upload Screen After Scope Change



Match Results

Fifa 2013
FifaTourno R-1
Match winner: Fifa team 1
Fifa team 1 uploaded: 1 - 4
Fifa team 2 uploaded: 100000 - 0000
Dispute status: Dispute Logged

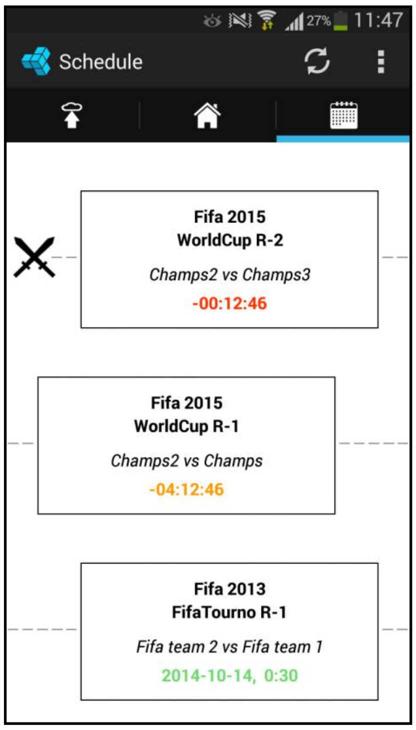
Fifa 2013
FifaTourno R-2
Match winner: Fifa team 3
Fifa team 2 uploaded: 2 - 3
Fifa team 3 uploaded: 3 - 2
Dispute status: Dispute Logged

Notifications

- Your team, Testers, has left the tournament: DotaOctober
2014-09-29 11:24

- Your team, Testers, has joined the tournament:

Figure 19: Home Screen After Scope Change



Fifa 2015
WorldCup R-2
Champs2 vs Champs3
-00:12:46

Fifa 2015
WorldCup R-1
Champs2 vs Champs
-04:12:46

Fifa 2013
FifaTourno R-1
Fifa team 2 vs Fifa team 1
2014-10-14, 0:30

Figure 20: Schedule Screen After Scope Change

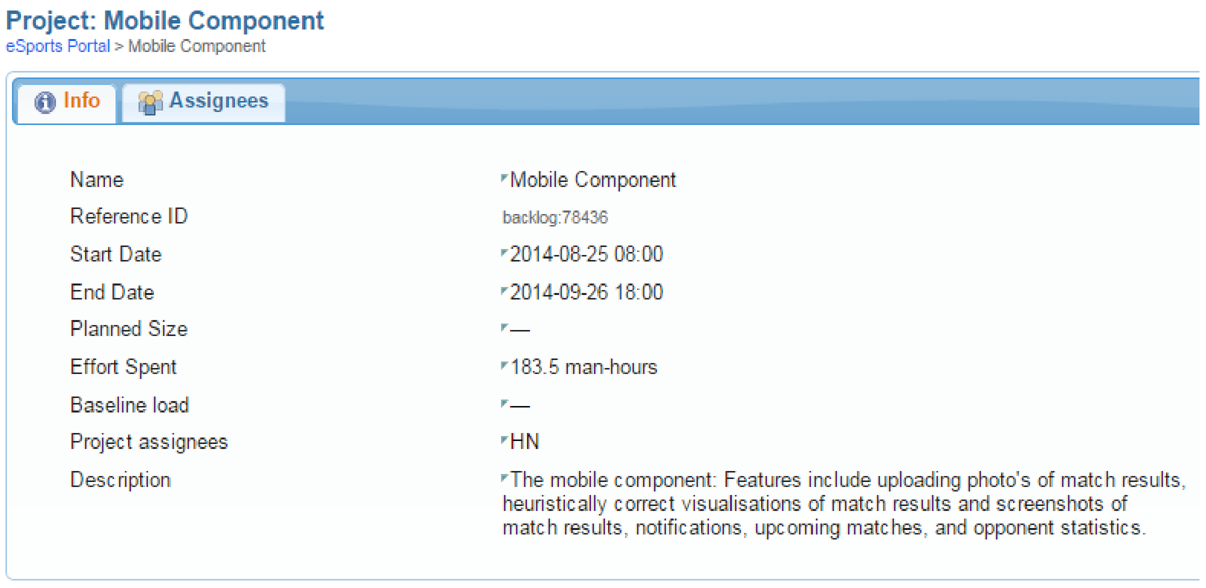
5. eSports Application and Web-service Implementation

5.1 Development Methodology: Scrum

Scrum methodology was chosen to manage the project cycle. Scrum methodology follows the agile development principle which is an iterative development methodology. However, Scrum methodology was not followed to its full potential. A meeting to report what was done and what will be done for the day was held electronically through a messenger system and not face to face. The meetings were also very irregular. Even though the overall project was not managed as well as it should be, the prototype iterations were completed with success. The whole cycle of choosing the priority features and implementing a working version allowed quick progress in implementing the features. Also the nature of the iterative approach allowed easy adaptation to changing requirements during the project.

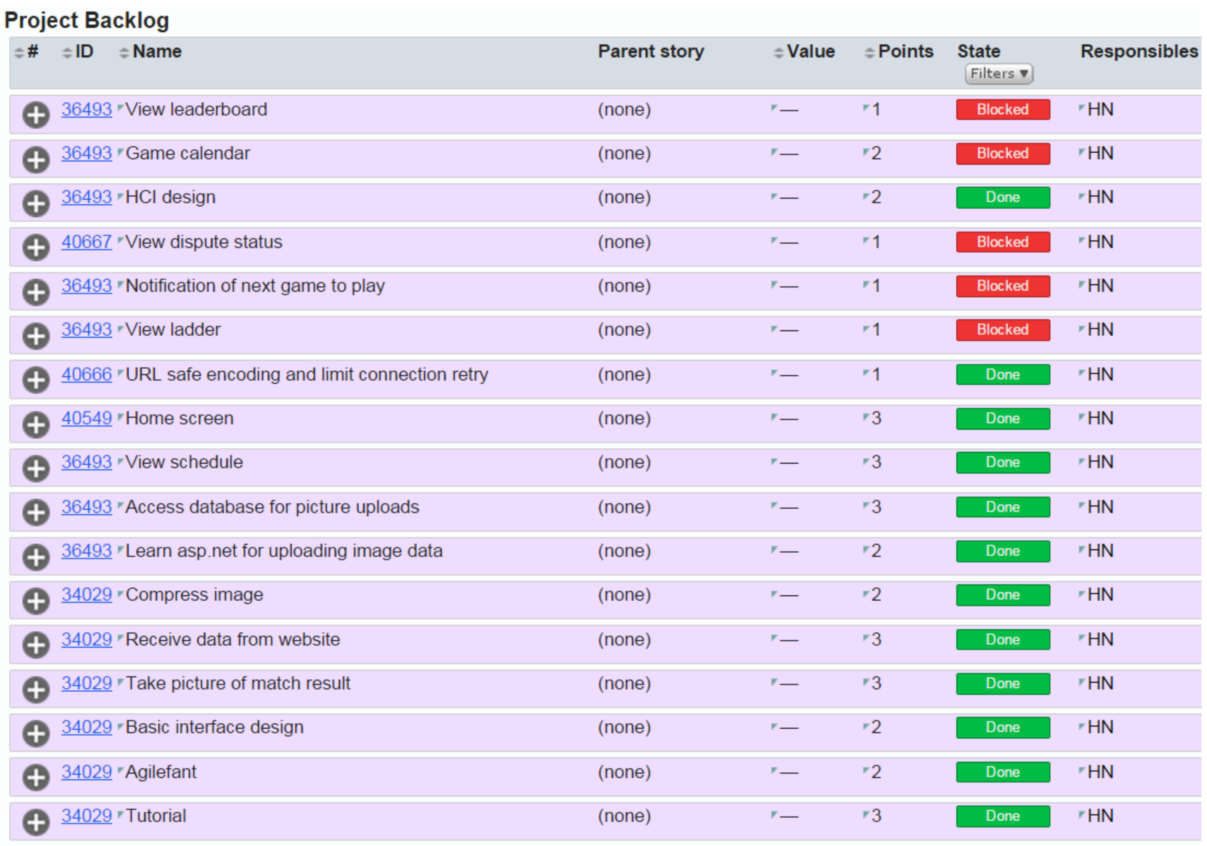
Agilefant, an online project management software, was used to manage the project cycle. Agilefant was chosen because it was one of the free online software that could be found and among the free software it seemed to have the best user interface. Some of the functionalities of Agilefant included shared iteration and sprint board, adding stories and tasks, workload chart, burn-down chart and work progress chart. Agilefant was helpful in checking the project progress through the burn-down chart and estimating the time required to finish certain tasks. The advantage of being online was that all the team members could check on each others' progress throughout the project and the risk of losing the data was low. It proved to be a useful tool in managing the software development project.

Figures 21 - 23 show the main features of Agilefant:



Name	Mobile Component
Reference ID	backlog:78436
Start Date	2014-08-25 08:00
End Date	2014-09-26 18:00
Planned Size	—
Effort Spent	183.5 man-hours
Baseline load	—
Project assignees	HN
Description	The mobile component: Features include uploading photo's of match results, heuristically correct visualisations of match results and screenshots of match results, notifications, upcoming matches, and opponent statistics.

Figure 21: Agilefant Project Front Page



Project Backlog

#	ID	Name	Parent story	Value	Points	State	Responsibles
⊕	36493	View leaderboard	(none)	—	1	Blocked	HN
⊕	36493	Game calendar	(none)	—	2	Blocked	HN
⊕	36493	HCI design	(none)	—	2	Done	HN
⊕	40667	View dispute status	(none)	—	1	Blocked	HN
⊕	36493	Notification of next game to play	(none)	—	1	Blocked	HN
⊕	36493	View ladder	(none)	—	1	Blocked	HN
⊕	40666	URL safe encoding and limit connection retry	(none)	—	1	Done	HN
⊕	40549	Home screen	(none)	—	3	Done	HN
⊕	36493	View schedule	(none)	—	3	Done	HN
⊕	36493	Access database for picture uploads	(none)	—	3	Done	HN
⊕	36493	Learn asp.net for uploading image data	(none)	—	2	Done	HN
⊕	34029	Compress image	(none)	—	2	Done	HN
⊕	34029	Receive data from website	(none)	—	3	Done	HN
⊕	34029	Take picture of match result	(none)	—	3	Done	HN
⊕	34029	Basic interface design	(none)	—	2	Done	HN
⊕	34029	Agilefant	(none)	—	2	Done	HN
⊕	34029	Tutorial	(none)	—	3	Done	HN

Figure 22: Agilefant Project Backlog

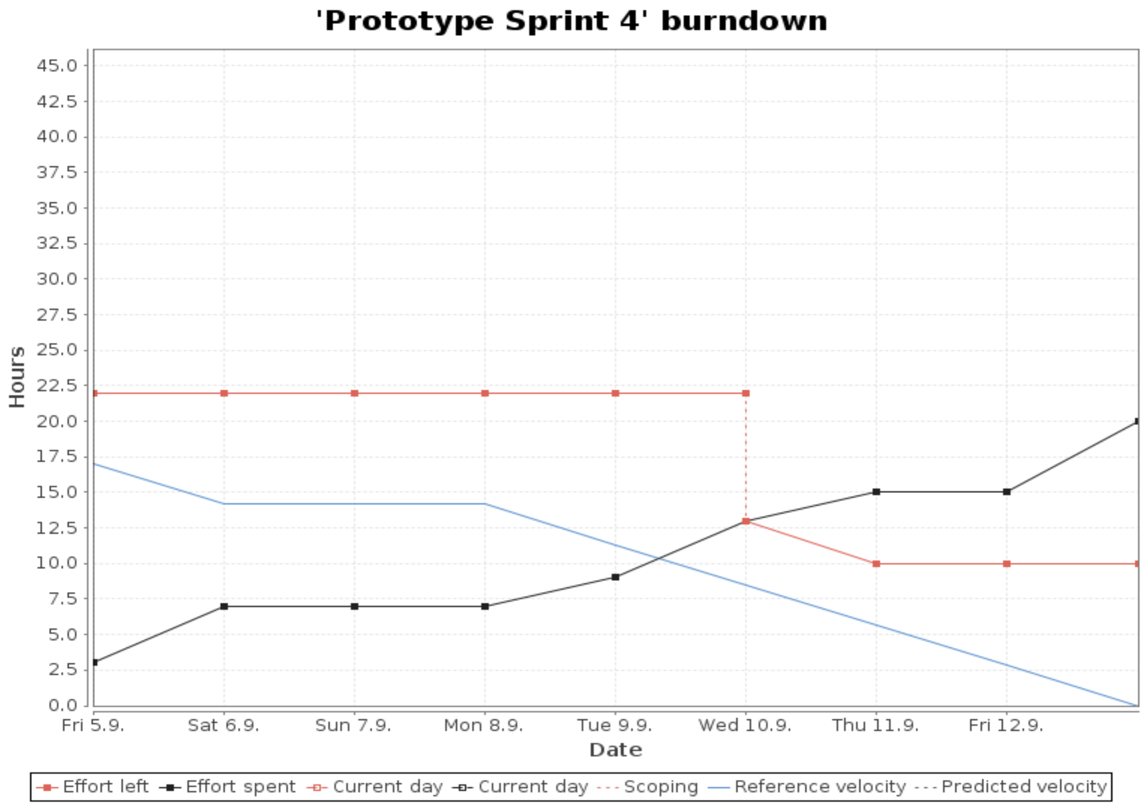


Figure 23: Agilefant Burndown Chart

The implementation of the mobile development was done in three iterations: First Prototype, Second Prototype and Final Prototype. Each prototype was broken down into two sprints where each sprint consisted of seven days.

5.1.1 First Prototype – Using Android and Phone Camera

Sprint 1

First sprint was going through Android tutorials and familiarizing the Android environment. It started with the basic "Hello world" examples and moved onto the life cycle of the Android Activities, managing Views, getting user input and displaying output, containers and layouts, creating new Activities, communication between Activities, internal and external storages, using default camera application and accessing resources.

Sprint 2

Second sprint was implementing the camera function. Basic application was made to take a picture and display it on the Image View. Image compression and data persistence had to be handled to limit the memory usage. The first prototype was this simple application that can take pictures and display the pictures without heavy memory usage.

5.1.2 Second Prototype – Basic Web-services

Sprint 3

In the third sprint, AsyncTask, which runs in the background thread, was implemented to download text and image files from the Android tutorial website. Default HttpClient was used to open the URL connections. Receiving and parsing of JSON data string was implemented as well.

Sprint 4

Fourth sprint was where the project went through some changes. Initially this sprint involved connection to the database of eSports website but due to the missing web-service from the eSports website, this sprint was changed to implement the web-service. With the updated sprint, ASP.Net and the WCF services had to be learnt first to create a web-service within the eSports website. After learning how to setup a Web.config file of the eSports website, a web-service was created but the connection to the database was not set up. At the end of the fourth sprint, basic web-service and the eSports application which could request and parse JSON data and the image files were presented as the second prototype.

5.1.3 Final Prototype – Frontend and Backend Completion

Sprint 5

Fifth sprint involved the implementation of database connection of the web-service, database query from the web-service and eSports application that could send and receive data through the web-service.

Sprint 6

The last sprint was implementing the main interface of the eSports application and adding refresh and logout features. Different font sizes and highlighting of texts were tested. Heuristic evaluation was done by two HCI experts and the interface was updated.

5.2 eSports Application Implementation

This section will discuss the implementation of the eSports application in more detail.
It will cover the development environment, interface layout, layout implementation, class description, and system workflow.

5.2.1 Development Environment

As mentioned in the previous chapters, eSports application was developed as an Android application. Eclipse IDE with the Android plug-in was used to develop the application. Android applications are developed using the Java language. The minimum software development kit (SDK) was set to application programming interface (API) level 17 and the targeted SDK was API level 19.

5.2.2 Interface Layout

Android offers Views which are used to define the visual structure for a user interface. Views are items such as Button, CheckBox, ProgressBar, TextView, etc. A View can be comprised of multiple other Views. Layout is a special View that acts as a container to hold other Views [8].

Linear Layout was used as the main container for structuring the eSports application interface. Linear Layout gives the option of placing the inner components vertically or horizontally. This was very useful in creating subsections within sections.
ScrollView was used to hold the entire layout for a scrollable interface. ScrollView only allows one child layout. Therefore all the sub-layouts had to be wrapped by one parent layout as seen in Figures 24 – 27.

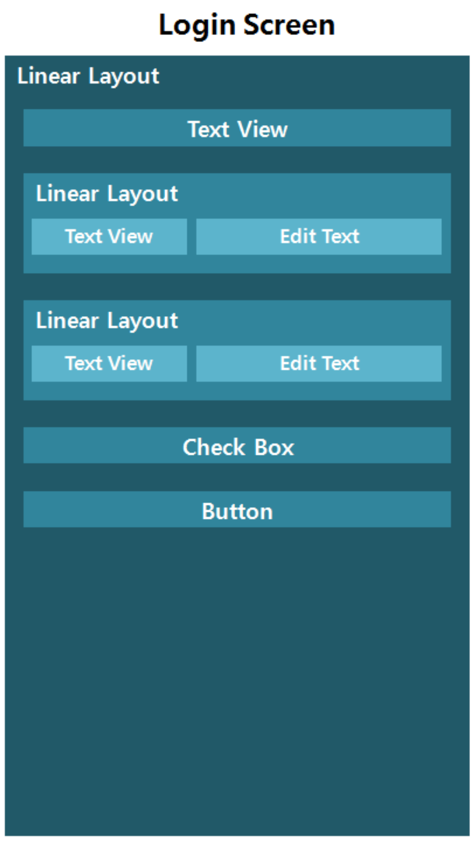


Figure 24: Login Layout

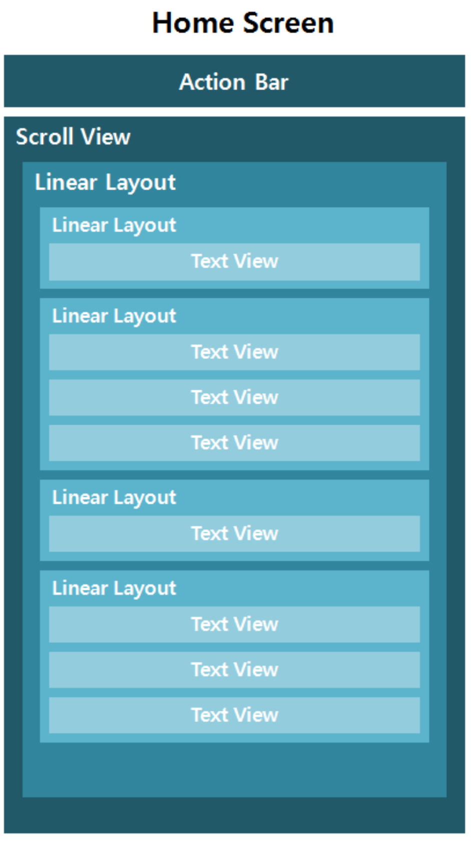


Figure 25: Home Layout

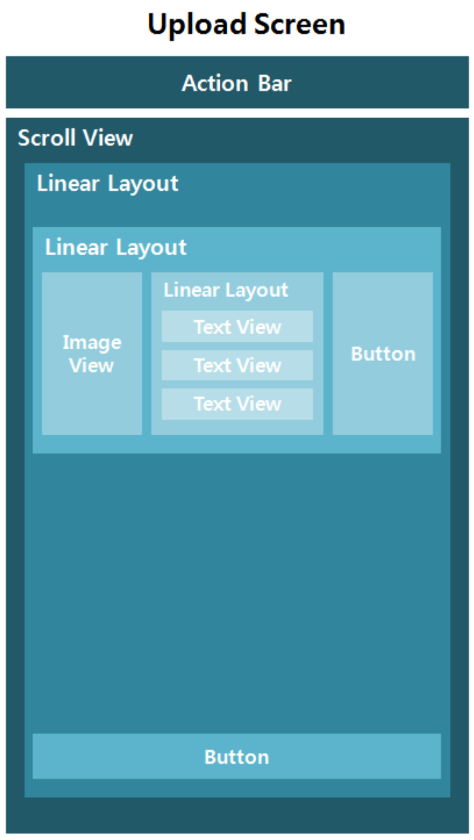


Figure 26: Upload Layout

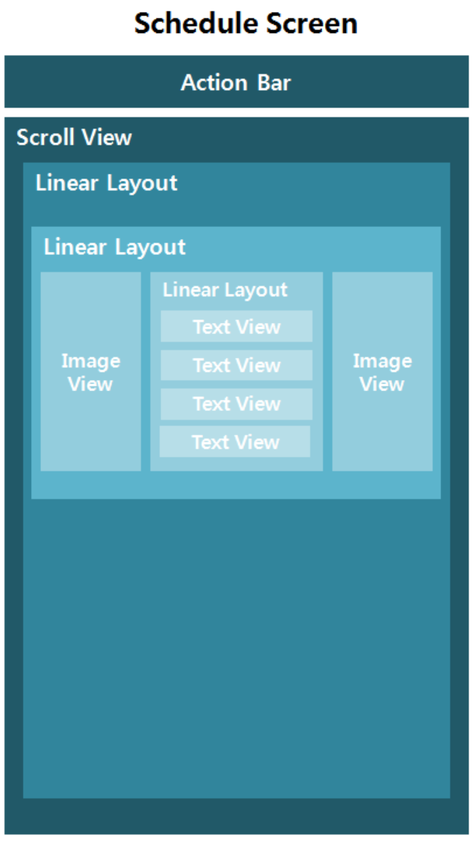


Figure 27: Schedule Layout

5.2.3 Layout Implementation

Layouts can be declared in XML or instantiated at runtime. Android framework gives the flexibility to use either or both methods [8]. All three methods were used to implement the layout in the eSports application. The Login screen's layout was fully done in the XML side, Home screen's skeleton layout with the heading was done in XML and the contents were instantiated at runtime. The Upload screen and Schedule screen were instantiated fully at runtime. Declaring the layouts was much easier and faster in XML with the help of the graphical visualization of the user interface and the drag and drop feature offered in Eclipse. Initially, layouts for all the screens were done using the graphical visualization feature. This helped to get an idea of how the layouts should be instantiated at runtime and the sizes of sub-layouts and Views could be configured easily.

Runtime implementation was necessary for the Upload and Schedule screen due to the dynamic nature of the two screens. Both screens show a dynamic list of items: list of past matches for Upload screen and list of upcoming matches for Schedule screen. This meant that the layouts could not be preconfigured in XML. Only when the total list of items to show was determined, could the layouts be set up. Home screen required partial runtime implementation for displaying the list of match results and list of notifications.

Although the nature of the information to display complicated the implementation of the layouts both programmatically and graphically, it aided the understanding of the Android layout framework and View components in more depth.

5.2.4 Class Description and Runtime Process

A total of nine classes were created for the eSports application. Among the classes, three were Activity classes with one utility class and three Fragment classes with two utility classes.

LoginActivity.java

This is the Android launcher class for eSports application. This class creates the first screen of the eSports application which is the login screen. It uses default EditText to get user-name input and a password protection EditText to get user password. The password protection EditText hides the text while inputting.

AsyncTask class is implemented as an inner class of LoginActivity class to send user credentials to the eSports web-service. AsyncTask class is used for running short operations in the background thread and upon completing a task the result can be published on the UI thread [4]. Four methods can be overridden in the AsyncTask: onPreExecute(), doInBackground(), onProgressUpdate() and onPostExecute().

In the onPreExecute() method, a progress bar gets started until the AsyncTask finishes. While the progress bar is running, HttpClient [3] is executed in the doInBackground() method to establish connection to the eSports web-service. A direct URL address with the IP address of the server hosting the web-service is used for establishing this connection. All the sending and receiving of the data between the application and the web-service are handled through HttpClient. Once the communication between HttpClient and eSports web-service terminates, onPostExecute() method publishes the result to the UI and the progress bar is dismissed. All data communication between the eSports application and the eSports web-service is handled in a similar manner.

When the user credentials are verified, LoginActivity launches the DownloadActivity through the use of intent. Intent provides for late runtime binding between codes in different applications [7]. It is mainly used for launching and sending simple data to the new Activity.

DownloadActivity.java

DownloadActivity creates an in-between loading screen when users navigate from the login screen to the home screen. The main functionality is to display a loading page while in the background, the match results, the match schedule and the notifications that are registered to the username get downloaded. The data is then stored separately in the internal storage as byte files. When all the data are downloaded and saved, SlideMenuActivity is launched.

SlideMenuActivity.java

SlideMenuActivity class is the host Activity for the HomeFragment, the UploadFragment and the ScheduleFragment. Listeners are implemented in this class for each mentioned Fragment to communicate between the Fragments and the Activity. Fragment represents a portion of user interface in an Activity and can be combined to create a multi-pane UI [5]. ActionBar [2] is used to create tab menu options. The refresh button and the overflow button that holds the logout button are also created with ActionBar. When the Fragment requests data, the corresponding data file saved from DownloadActivity is recalled and the data is sent back to the Fragment.

MenuPagerAdapter.java
MenuPagerAdapter is a utility class for SlideMenuActivity. This class dynamically creates the Fragments that are hosted by the SlideMenuActivity and connects the Fragments in a horizontal pane. This allows the user to change to different menus by swiping the screen sideways.

HomeFragment.java
HomeFragment creates the home screen of the eSports application which is the first screen that gets displayed after the loading page. It displays the recent match results the user has played. The match result data is determined by the uploading of the scores from the two teams that have finished a match. If the scores uploaded by both teams do not match, a dispute is logged and will be displayed under the match result. Notifications concerning the user will be displayed below the match result section.

UploadFragment.java
UploadFragment creates the upload screen which is left of the home screen. This can be accessed by swiping to the right from the home screen or by touching the upload menu in the ActionBar. This Fragment handles the uploading of image files and scores.

There were two main challenges during the implementation of this Fragment. The first challenge was to handle the memory usage of the eSports application. Modern day mobile devices are powerful with good amount of memory and cache size but this is still not enough to handle high quality images at runtime. The loading and displaying of images can be handled without any concern but when the image has to be kept in memory and reloaded to the UI, the image size far exceeds the allocated memory size. This could be handled in two ways, either by reducing the quality of the image thereby reducing the memory required to hold the image, or by saving the image to an internal storage and reading it back into the memory when required. The former approach was taken for eSports application. The main idea behind displaying captured image is to give feedback to the user that the image capture was successful. To achieve this task, image quality is not as important. Also, the size of the ImageView to hold the image is much smaller than the original image which means the reduced quality of the image will not be noticeable to the user at all.

The second challenge was handling the data persistence. The design of the eSports application allows for a set portrait view for the entire application. This meant that the screen orientation changes did not have to be dealt with and to enforce the portrait view, all screens in eSports application were set to have no screen orientation changes. However, the orientation changes outside of the eSports application cannot be controlled and due to the unhandled data persistence in case of orientation change, UploadFragment resulted in data loss errors. Figure 28 shows the orientation change that was triggered from default camera application and the data loss resulting from the change.

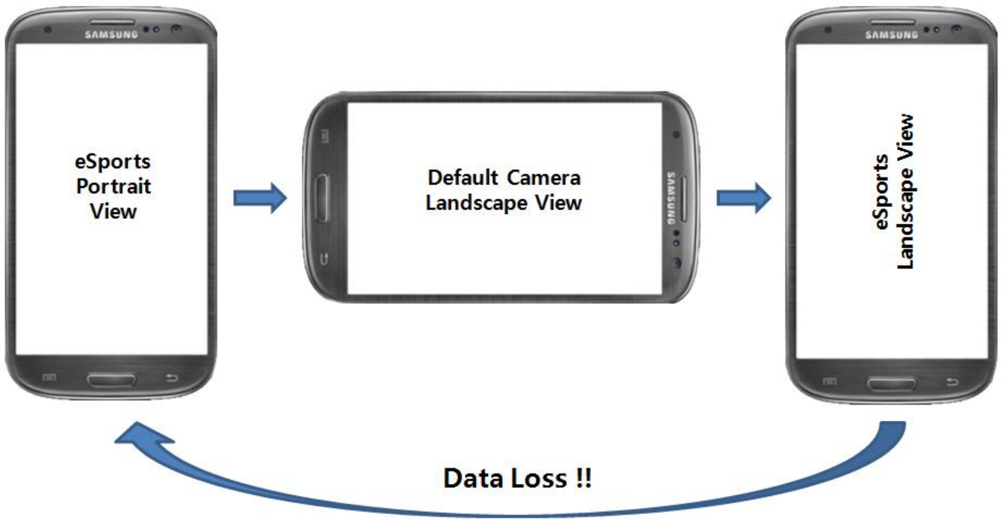


Figure 28: Orientation Change from Default Camera Application

When Android applications go through an orientation change, the Activity or Fragment that created the screen gets destroyed and recreated. Any unsaved data during this cycle will be lost.

Again there were two ways to solve this issue: by saving the data to the internal storage before launching the default camera application or using onSaveInstanceState() method [10]. Android offers a method called onSaveInstanceState() to deal with these kind of situations. onSaveInstanceState() saves the data in the cache and retrieves it on recreation of the destroyed Activity or Fragment. Both ways had to be used for eSports application due to some of the data to be saved being an image. This image could not be resized or reduced in quality because the original image had to be saved for the server. The solution was to save the image as a byte file in the internal storage and use onSaveInstanceState() for the rest of the data.

ScheduleFragment.java
ScheduleFragment creates the schedule screen. This is placed on the right of the home screen and can be accessed by swiping to the left from home screen. It displays the upcoming matches with a countdown timer when the match is less than 24 hours away. If the upcoming matches are more than 24 hours away, the date and time of the match are displayed. The box that shows the upcoming match information is set to be clickable. When the box is pressed, the opponent team's win/lose/draw is shown as a pop-up.

The main challenge in the implementation was the time calculation. From the eSports web-service, all the matches in which the user is involved (including past and future matches) with their date and time are sent to the eSports application. This data is then categorized to past and upcoming matches by checking the current date and time of the system against the match date and time. Calculating time differences is straight forward but calculating differences between two dates were a bit more challenging. The problem was taking into account leap years and checking which month it is to know whether the month has 30 days or 31 days. Converting the two dates to Gregorian days and getting the difference was tried but could not get an accurate result. After couple of unsuccessful attempts, open source library called Joda time was used to do the calculation. Joda time library offers daysBetween() method for getting the days in between two dates.

BitmapHandler.java
BitmapHandler is a utility class for UploadFragment. It handles the image resizing.

MatchTimeHandler.java
MatchTimeHandler is a utility class for ScheduleFragment. All the calculations of upcoming and past match dates and times are done in this class. User time zones are factored into the calculations to output the correct time for the users.

Figure 29 shows the overall class relations.

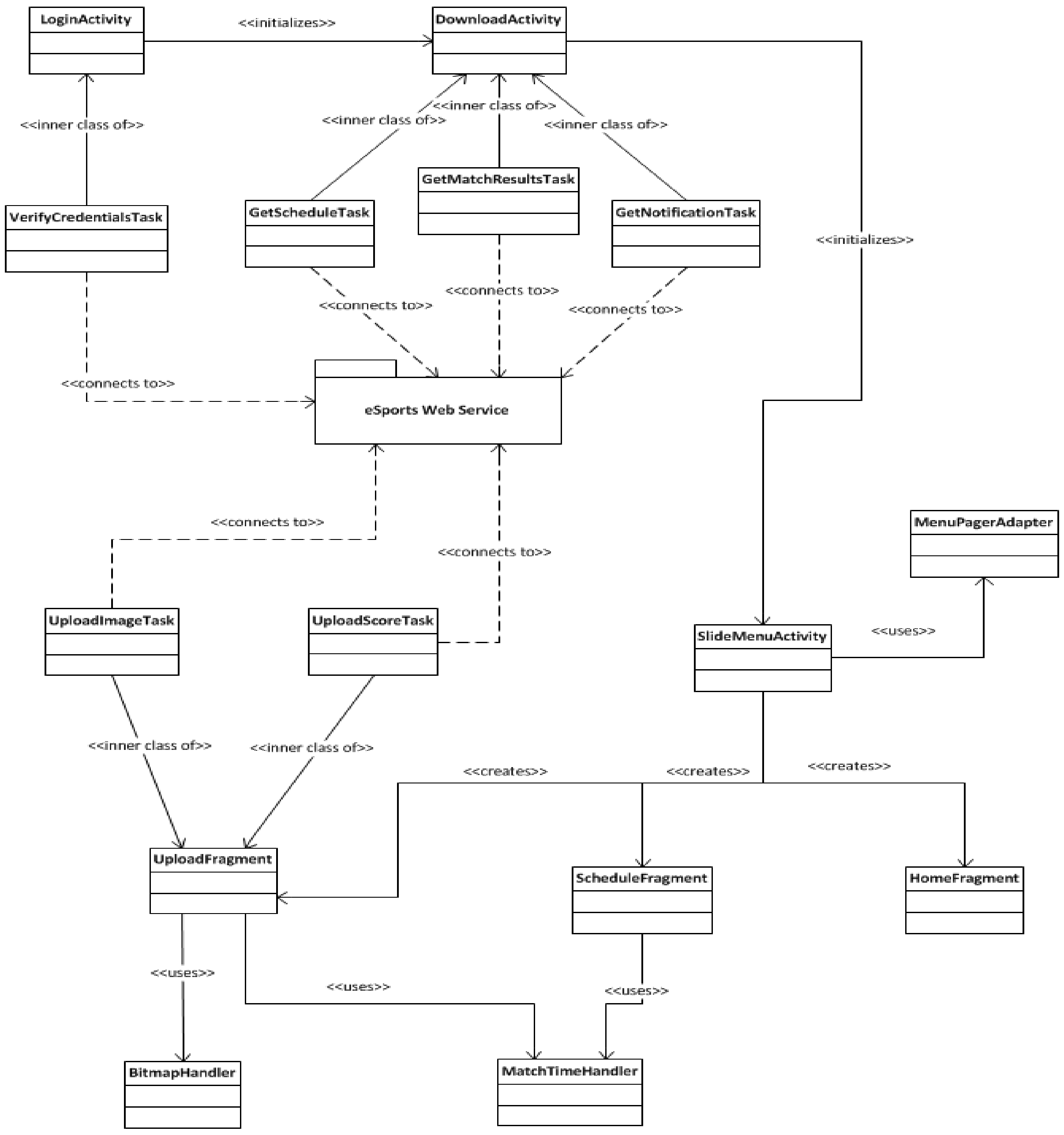


Figure 29: Class Diagram

5.2.5 System Workflow

Figure 30 shows the sequence diagram for simple view operation in eSports application. The user logs in, swipes right to see upload screen, swipes left twice to see schedule screen and presses back button to exit.

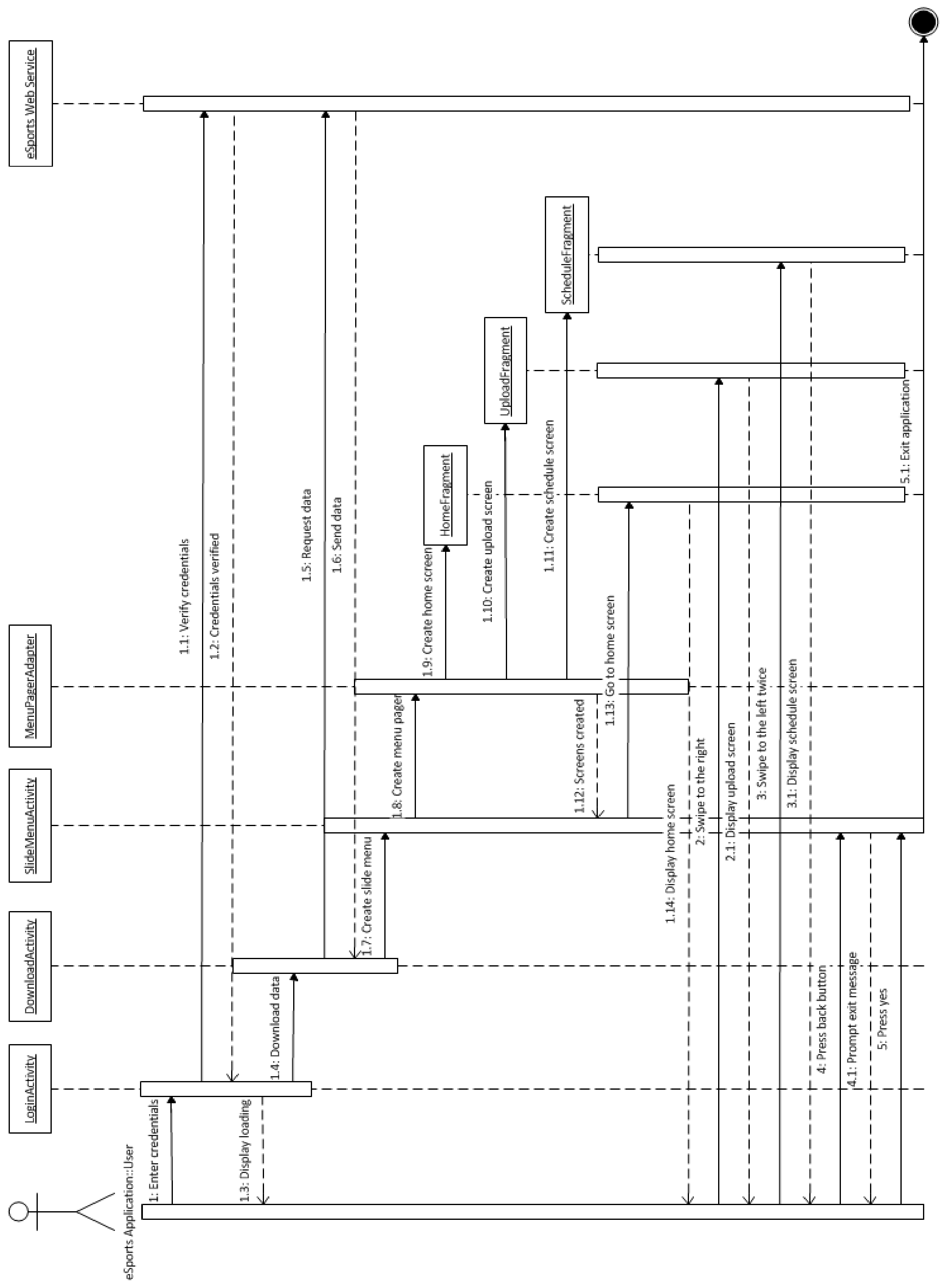


Figure 30: Login and Logout Sequence Diagram

5.3 eSports Web-service Implementation

5.3.1 Development Environment

eSports web-service was created with Microsoft Visual Studio 2013 (MSVS) and it uses the C# language and the web-services Description Language (WSDL). There are two main ways of creating the web-service in MSVS: using the traditional ASMX, which is a straight forward web-service with SOAP protocol, or using the WCF Service which is a unified framework that combines the web-service, .Net Remoting, Enterprise Services and Message Queuing under a unified programming model [23]. WCF Service was used to create the eSports web-service due to the support for REST protocol and the added security and reliability.

5.3.2 Class Descriptions

eSports web-service consists of one interface, two classes and one XML file. The interface is written in WSDL and the two classes are written in C#.

IAndroidRestService.cs

IAndroidRestService is the Service Contract interface for the eSports web-service. Service Contract declares Operation Contract that allows URL connections to the eSports web-service. Each Operation Contract defines the type of data allowed through the URL connection and the response format which is sent back to the client.

Data Contract classes are defined in the IAndroidRestService as well to model the eSports database tables into classes.

AndroidRestService.svc.cs

AndroidRestService implements the Service Contract of the IAndroidRestService. All the client requests that come through the URL connections are handled through the corresponding method declared in AndroidRestService.
The URL connection is in the form:
'http://*webserviceIPaddress*/eSportsPortal/AndroidRestService.svc/*methodname*/*arguments*/'

When the eSports application sends the user credentials to the eSports web-service, the URL looks like the following:
'http://192.168.0.1/eSportsPortal/AndroidRestService.svc/Login/cg3f4622e1e5dh83fhgee44/'

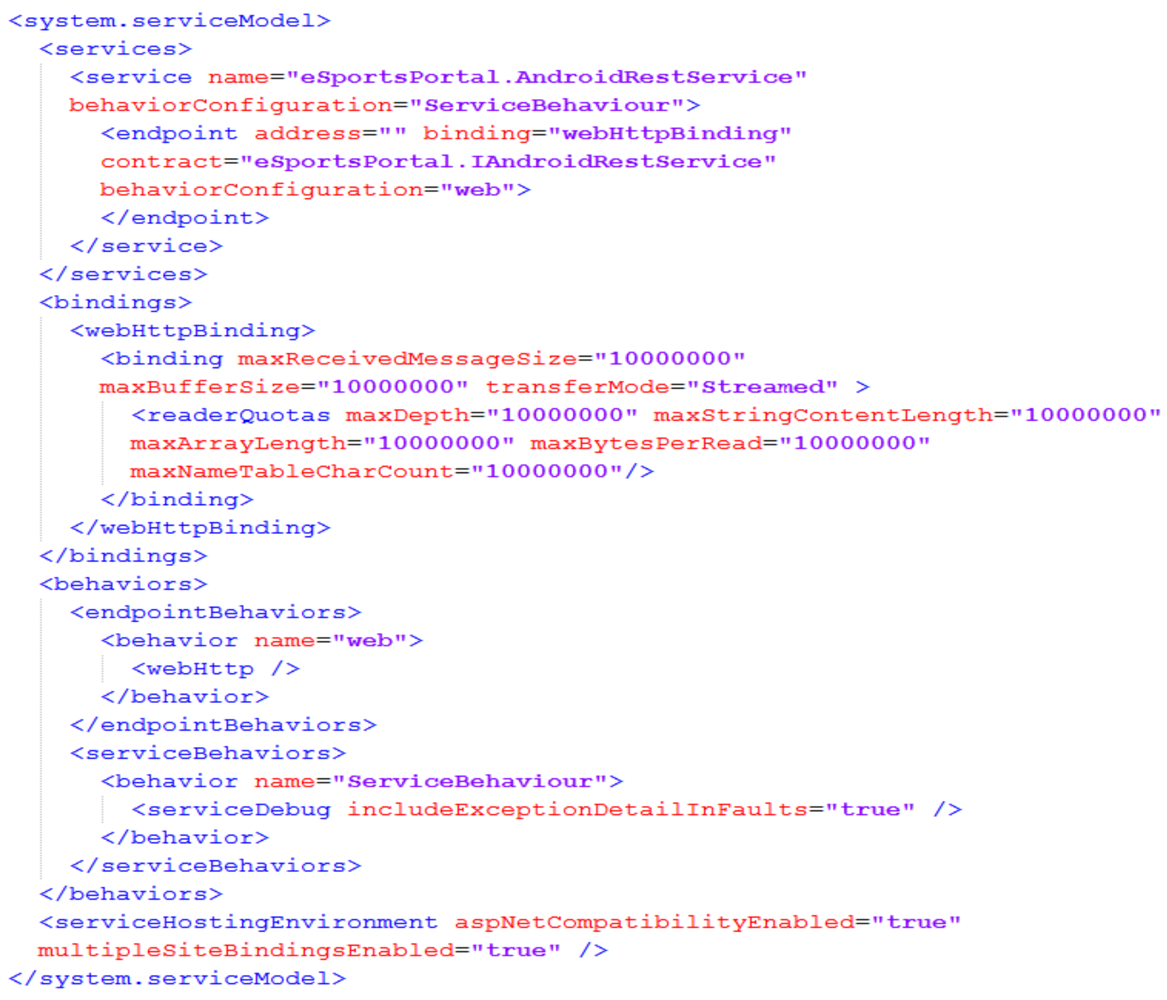
The Login method in eSports web-service will check the credentials against the database and return a response back to the eSports application. A database connection string is used to access the eSports database and the queries for the client requests are formulated within the methods and the data is retrieved, updated or inserted to the database.

All the following methods receive and send data in a similar manner: GetNotification(), GetMatchResults(), GetSchedule(), GetUloadedImage(), PostImage(), and PostScore().

MultipartParser.cs
MutipartParser is a utility class for AndroidRestService. This class parses the multipart data sent from the eSports application. The multipart data consists of a header followed by the contents. The only multipart data from the eSports application is the image file, so this parser is customized specifically to parse the multipart image file. Regular expressions are used to extract file name and file type from the header, and the contents are saved to a file. The image file is stored in the Content folder of the Server and the file name is saved in the database. The Content folder is created if the folder is missing from the Server.

Web.config
Web.config is a XML file used to configure the eSports website which is created using ASP.NET. Additional code is needed to be included in this file to host the eSports web-service in conjunction with the eSports website. Also, the REST protocol and the byte size for incoming streams of image file are configured in this file.

Figure 31 shows the Web.config file. Binding type is webHttpBinding which uses REST protocol and the transfer mode is set to be streamed with maximum buffer size of 10MB in both sending and receiving data. An average photo size is about 2MB, therefore, the buffer was set to 10MB to ensure that it does not run out of space.



```
<system.serviceModel>
  <services>
    <service name="eSportsPortal.AndroidRestService"
    behaviorConfiguration="ServiceBehaviour">
      <endpoint address="" binding="webHttpBinding"
      contract="eSportsPortal.IAndroidRestService"
      behaviorConfiguration="web">
      </endpoint>
    </service>
  </services>
  <bindings>
    <webHttpBinding>
      <binding maxReceivedMessageSize="10000000"
      maxBufferSize="10000000" transferMode="Streamed" >
        <readerQuotas maxDepth="10000000" maxStringContentLength="10000000"
        maxArrayLength="10000000" maxBytesPerRead="10000000"
        maxNameTableCharCount="10000000"/>
      </binding>
    </webHttpBinding>
  </bindings>
  <behaviors>
    <endpointBehaviors>
      <behavior name="web">
        <webHttp />
      </behavior>
    </endpointBehaviors>
    <serviceBehaviors>
      <behavior name="ServiceBehaviour">
        <serviceDebug includeExceptionDetailInFaults="true" />
      </behavior>
    </serviceBehaviors>
  </behaviors>
  <serviceHostingEnvironment aspNetCompatibilityEnabled="true"
  multipleSiteBindingsEnabled="true" />
</system.serviceModel>
```

Figure 31: Web.config File

5.3.3 Implementation Challenges

The main challenge in implementing the eSports web-service was the learning of ASP.NET and WCF Service in a short time frame. Only the basic knowledge of handling the eSports web-service was acquired and due to the lack of knowledge in ASP.NET, two problems were encountered.

The first problem was hosting the eSports web-service separately from the main website. When the eSports web-service was hosted separately, it could not access the database. The error was unauthorized access and because the eSports website was created without the web-service, it was difficult to change the access settings.

The second problem was using Windows authentication for verifying credentials and securing the URL access. This problem was linked to the first problem because in order for the Windows authentication to work, eSports web-service had to be hosted separately. This would not have been a problem if the mobile application platform was Windows. Android does not have support for Windows authentication and therefore the security had to be dealt with separately from the eSports website, but due to the restricted database access this could not be done. Both the eSports website and the eSports application therefore had to use forms authentication for security instead of the Windows authentication.

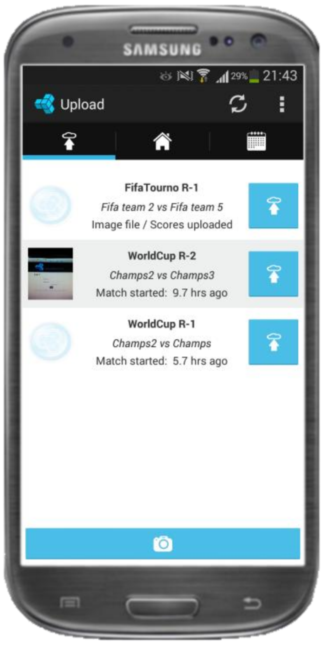
6. Heuristic Evaluation and User Testing

Two HCI experts and five users participated in the evaluation and user testing process. The eSports application was installed in a Samsung Galaxy S3 for both the evaluation and the user testing.

6.1 Heuristic Evaluation and Interface Change

Two HCI experts were consulted separately for a heuristic evaluation. One of the experts with a PhD in Computer Science was consulted first and then the expert with a MSc in Information Systems was consulted next.

The following Figures show Upload screen, Home screen, and Schedule screen of the eSports application before the heuristic evaluation.

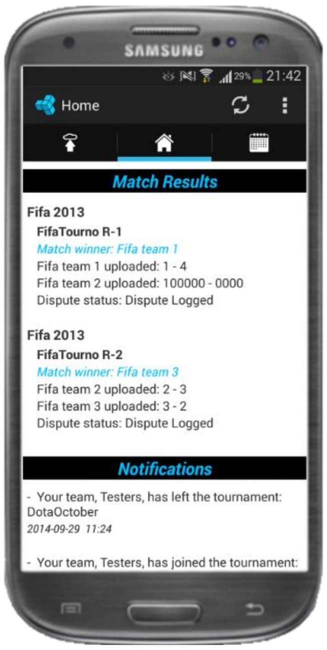


FifaTourno R-1
Fifa team 2 vs Fifa team 5
Image file / Scores uploaded

WorldCup R-2
Champs2 vs Champs3
Match started: 9.7 hrs ago

WorldCup R-1
Champs2 vs Champs
Match started: 5.7 hrs ago

Figure 32: Upload Screen Before Evaluation



Match Results

Fifa 2013
FifaTourno R-1
Match winner: Fifa team 1
Fifa team 1 uploaded: 1 - 4
Fifa team 2 uploaded: 100000 - 0000
Dispute status: Dispute Logged

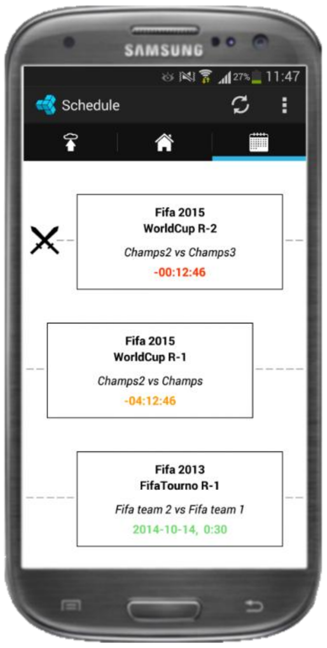
Fifa 2013
FifaTourno R-2
Match winner: Fifa team 3
Fifa team 2 uploaded: 2 - 3
Fifa team 3 uploaded: 3 - 2
Dispute status: Dispute Logged

Notifications

- Your team, Testers, has left the tournament: DotaOctober
2014-09-29 11:24

- Your team, Testers, has joined the tournament:

Figure 33: Home Screen Before Evaluation



Fifa 2015
WorldCup R-2
Champs2 vs Champs3
-00:12:46

Fifa 2015
WorldCup R-1
Champs2 vs Champs
-04:12:46

Fifa 2013
FifaTourno R-1
Fifa team 2 vs Fifa team 1
2014-10-14, 0:30

Figure 34: Schedule Screen Before Evaluation

6.1.1 Upload Screen

Evaluation

It was pointed out in the upload screen that an already uploaded picture was not displayed in the image view. When the image was uploaded to the server, the upload screen only had text indication that the image was uploaded but did not display the uploaded picture. Also, once the application was refreshed or re-launched, all the previously displayed pictures were not displayed.

Another comment was the distance between the camera button and the upload button. When there is only one match result to upload, the distance between the upload button and camera button was too big and a gap appeared in the middle of the screen. Consistency in showing the heading of the screen on top of every page was suggested.

Implementation
The missing pictures could not be implemented in time for the final prototype.
The distance between the camera button and the upload button was addressed by centering the list of matches but some errors were encountered and could not be fixed in time so the old version was kept. Heading of the screen was added to be consistent with the match result screen.

6.1.2 Home Screen

Evaluation
It was suggested to remove the notification section and add it as an extra button on top of the screen. The reasoning was to avoid too much scrolling in one screen. The match results were criticized for having only text and the layout of the whole match result was suggested to be changed to a card layout. A question was raised about the refresh button and why it is not automated.

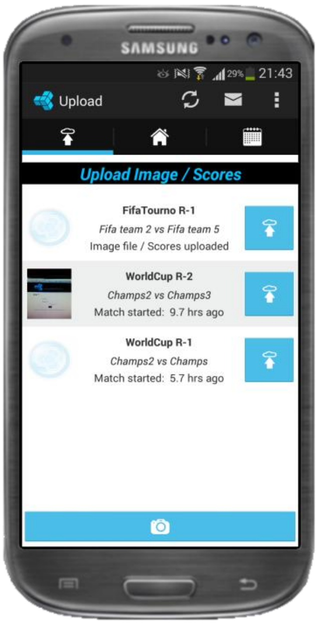
Implementation
Notification was removed and added as an action button in the top of the screen next to the refresh button. When the notification button is pressed the notification shows as a popup. Few icons were added to show win, loss, draw and dispute. The layout of the match result was changed as well to be shown in a box to emulate the card layout. Implementation of the card layout could not be done within the given time frame. The automation of the refresh screen could not be implemented due to the code needing to be re-factored. A new button was added as an extra feature to show uploaded picture from both teams in case of dispute.

6.1.3 Schedule Screen

Evaluation
Again it was suggested to change the layout to card layout so that it will be consistent with the match result screen and the display of information will be more user friendly. The dashed lines were considered to be confusing and the battle icon was considered as unnecessary in showing that there is a next match. The minus in front of count-down timer was asked to be removed because it gave the wrong idea that the time for match has already been passed. When the text within the box was pressed, the opponent team’s past matches were displayed. This was not intuitive at all and adding a button to show the opponent team’s past matches was suggested.

Implementation
The layout was changed similar to the match results screen using borders to emulate the card layout feel. A button was added to show the opponent team’s past matches. Heading of the screen was added to be consistent with the match result screen.

Figures 35 - 40 show the screenshots of the final prototype.



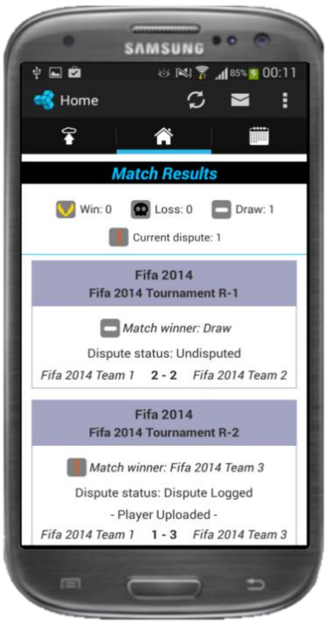
Upload Image / Scores

FifaTourno R-1
Fifa team 2 vs Fifa team 5
Image file / Scores uploaded

WorldCup R-2
Champs2 vs Champs3
Match started: 9.7 hrs ago

WorldCup R-1
Champs2 vs Champs
Match started: 5.7 hrs ago

Figure 35: Upload Screen Final Prototype



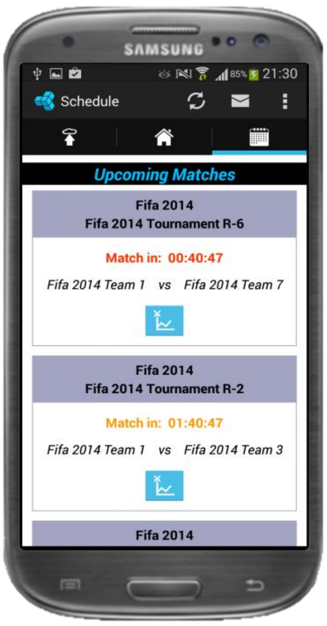
Match Results

Win: 0 Loss: 0 Draw: 1
Current dispute: 1

Fifa 2014
Fifa 2014 Tournament R-1
Match winner: Draw
Dispute status: Undisputed
Fifa 2014 Team 1 2 - 2 Fifa 2014 Team 2

Fifa 2014
Fifa 2014 Tournament R-2
Match winner: Fifa 2014 Team 3
Dispute status: Dispute Logged
- Player Uploaded -
Fifa 2014 Team 1 1 - 3 Fifa 2014 Team 3

Figure 36: Home Screen Final Prototype



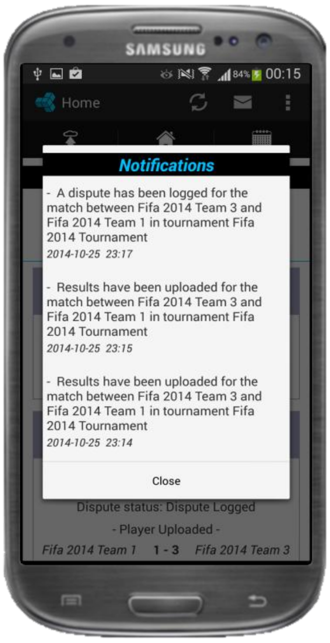
Upcoming Matches

Fifa 2014
Fifa 2014 Tournament R-6
Match in: 00:40:47
Fifa 2014 Team 1 vs Fifa 2014 Team 7

Fifa 2014
Fifa 2014 Tournament R-2
Match in: 01:40:47
Fifa 2014 Team 1 vs Fifa 2014 Team 3

Fifa 2014

Figure 37: Schedule Screen Final Prototype



Notifications

- A dispute has been logged for the match between Fifa 2014 Team 3 and Fifa 2014 Team 1 in tournament Fifa 2014 Tournament
2014-10-25 23:17

- Results have been uploaded for the match between Fifa 2014 Team 3 and Fifa 2014 Team 1 in tournament Fifa 2014 Tournament
2014-10-25 23:15

- Results have been uploaded for the match between Fifa 2014 Team 3 and Fifa 2014 Team 1 in tournament Fifa 2014 Tournament
2014-10-25 23:14

Close

Dispute status: Dispute Logged
- Player Uploaded -
Fifa 2014 Team 1 1 - 3 Fifa 2014 Team 3

Figure 38: Notification Pop-up Final Prototype

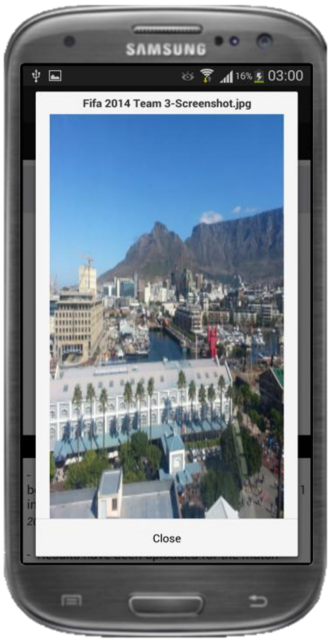
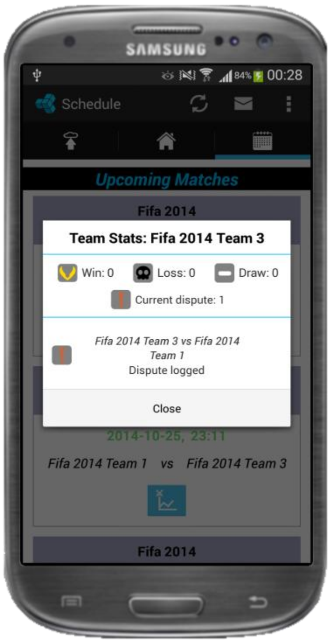


Figure 39: Screenshot Pop-up Final Prototype



Upcoming Matches
Fifa 2014

Team Stats: Fifa 2014 Team 3
Win: 0 Loss: 0 Draw: 0
Current dispute: 1
Fifa 2014 Team 3 vs Fifa 2014 Team 1
Dispute logged
Close

2014-10-25, 23:11
Fifa 2014 Team 1 vs Fifa 2014 Team 3
Fifa 2014

Figure 40: Statistics Pop-up Final Prototype

6.2 User Testing

Five users were asked to do various tasks with the eSports application. These five users were: iStore manager, Computer Science student, software tester, professional athlete and a music teacher. All five users were confident smart phone users. Four of the five users were familiar with eSports. A table with a set of tasks were given to the users and they were asked to complete the task using the eSports application and rate the difficulty of the task from a scale of 1 – 5 where 1 being extremely easy and 5 being extremely hard.

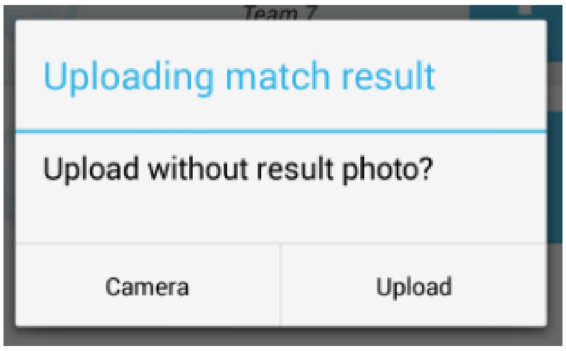
The following table presents the rating given by five users for each task:

Table 3: Task Difficulty Rating

Task	**User A**	**User B**	**User C**	**User D**	**User E**	**AVG**
1. Navigate to upload screen	1	1	1	1	2	1.2
2. Upload result photo and score	1	3	2	3	1	2
3. Find the latest match result	3	1	1	1	1	1.4
4. Find the match result with a dispute	1	2	3	2	1	1.8
5. View the opponent team's screenshot in case of dispute logged	4	1	1	1	3	2
6. Navigate to schedule screen	1	1	1	1	1	1
7. View which match will be played next	2	1	1	1	1	1.2
8. View the match statistics of the opponent team you are about to verse	2	1	5	1	3	2.4
9. View notification	1	5	3	1	1	2.2
10. Logout	1	1	2	1	1	1.2
11. Exit application	1	3	1	1	1	1.4
Total (range is 11 for all extremely easy, to 55 for all extremely hard)	18	20	21	14	16	17.8

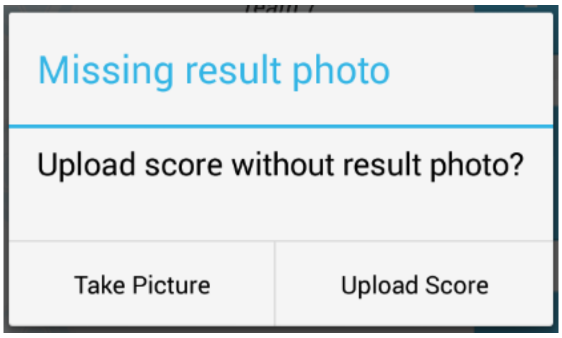
The test data is too small to make any conclusions but it is worth noting that tasks 2, 5, 8, and 9 seemed to be the hardest task for the users. One of the comments from a user for task 2 (upload image and score) was that the wording of the popup in the upload screen was not as clear in asking whether to upload score only or take picture first. The wording of the popup was changed to inform the user better.

Popup before testing: Popup after testing:


Uploading match result

Upload without result photo?

Figure 41: Upload Pop-up Before User Testing

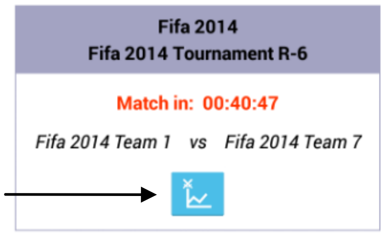

Missing result photo

Upload score without result photo?

Figure 42: Upload Pop-up After User Testing

Tasks 5, 8, and 9 all involved missing the respective buttons. Users that struggled commented that they did not recognize the buttons immediately and were checking out different screens to see if they missed any information. More user testing would be needed to confirm whether the difficulty was indeed failure to recognize the buttons or the placement and other factors that caused the confusion.

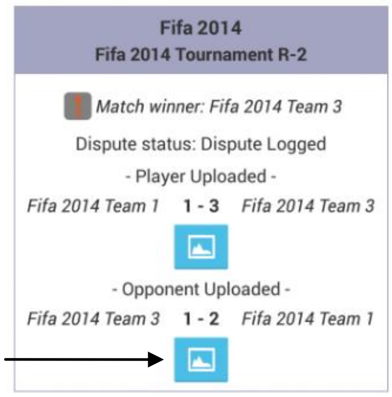
Buttons that some users missed:


Fifa 2014
Fifa 2014 Tournament R-6

Match in: 00:40:47

Fifa 2014 Team 1 vs Fifa 2014 Team 7

Figure 43: View Match Statistics Button


Fifa 2014
Fifa 2014 Tournament R-2

Match winner: Fifa 2014 Team 3

Dispute status: Dispute Logged

- Player Uploaded -

Fifa 2014 Team 1 1 - 3 Fifa 2014 Team 3

- Opponent Uploaded -

Fifa 2014 Team 3 1 - 2 Fifa 2014 Team 1

Figure 44: View Screenshot Button

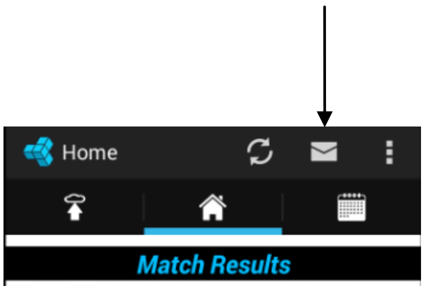


Figure 45: View Notifications Button

7. Future Work

Future work for the eSports application will involve implementing the feedback from heuristic evaluation and the missed features from the Requirements chapter.

The unimplemented feedback from the heuristic evaluation:

- Displaying already uploaded images in the Upload screen and displaying the image in full screen when the image is touched.
- Implementing card layout with expendable view for the entire interface.
- Automating the refresh button.

The missed features:

- View leader-board.
- View team ladder – fixtures displayed in a ladder (for knock-out stages).
- View/comment dispute process – show uploaded result photo and have comment thread under the photo.

In addition, the user testing should be done more thoroughly with more users.

Future work for the eSports web-service will involve switching to Windows authentication instead of forms authentication. This is to allow users to access the application with their Windows account rather than having to create a new username and password for the eSports application. To implement Windows authentication, the entire back-end architecture of the website and web-service would need to be changed.

8. Conclusion

This project turned out to be a valuable experience in software development. The theory of how the development should be carried out and the things that can go wrong in the development process were experienced firsthand. Many little challenges were encountered and the understanding of why managing software development is not an easy task grew during the course of this project.

The core functionality of taking a picture and uploading both the photo and the score was successfully implemented in the final prototype of eSports application. Other functionalities: displaying match results, displaying upcoming matches, notifications, and displaying disputed images and results were also implemented and tested to be working correctly. However, the scope of the features and quality of the features had to be scaled down to finish the project in time.

It was unfortunate that the HCI and interface design could not be done as well as it was planned. The main reason for this was the re-assigning of the web-service implementation from the website developer to the mobile developer. Much time had to be used in learning ASP.NET and WCF services to implement the web-service. Another cause of the time shortage was the under-estimated development time for some of the features in eSports application. The whole process of designing layouts, creating icons, formulating class architecture, and the actual code implementation took longer than expected.

The successful factor in the project was the iterative nature of the scrum methodology. Having sprints planned out and iteratively implementing the features allowed fast adaptation to changing scope and requirements. Following the scrum methodology ensured that a working feature was produced and any changes or add-ons to the existing feature could be handled easily.

Software development is a complicated process and developing a full software product for a client, Derivco, gave valuable lessons in managing time, understanding and capturing user requirements, defining scope, estimating task completion, designing for user satisfaction, evaluating and thorough testing and mitigating the changing requirements, which will contribute to a better management of the software development.

9. References

[1] Alonso, G., Casati, F., Kuno, H., Machiraju, V. 2004. Web Services. 123 – 149. Springer Berlin Heidelberg.

[2] Android Developers ActionBar
http://developer.android.com/guide/topics/ui/actionbar.html
[Accessed 07-10-2014]

[3] Android Developers Apache HttpClient
http://developer.android.com/reference/org/apache/http/client/HttpClient.html
[Accessed 06-10-2014]

[4] Android Developers AsyncTask
http://developer.android.com/reference/android/os/AsyncTask.html
[Accessed 06-10-2014]

[5] Android Developers Fragment
http://developer.android.com/guide/components/fragments.html
[Accessed 07-10-2014]

[6] Android Developers Iconography
http://developer.android.com/design/style/iconography.html
[Accessed 06-10-2014]

[7] Android Developers Intent
http://developer.android.com/reference/android/content/Intent.html
[Accessed 06-10-2014]

[8] Android Developers Layouts
http://developer.android.com/guide/topics/ui/declaring-layout.html
[Accessed 06-10-2014]

[9] Android Developers Providing Descendant and Lateral Navigation
http://developer.android.com/training/design-navigation/descendant-lateral.html
[Accessed 28-10-2014]

[10] Android Developers Recreating Activity
http://developer.android.com/training/basics/activity-lifecycle/recreating.html
[Accessed 07-10-2014]

[11] Charland, A., and Leroux, B. 2011. Mobile Application Development: Web vs. Native. *Communications of the ACM*. ACM, v.54. n.5. DOI=10.1145/1941487.1941504.

[12] Christensen, J. 2009. Using RESTful Web-Services and Cloud Computing To Create Next Generation Mobile Applications. OOPSLA '09: *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*. ACM, New York, 627-634. DOI=10.1145/1639950.1639958.

[13] Church, K., and Oliver, N. 2011. Understanding Mobile Web and Mobile Search Use in Today's Dynamic Mobile Landscape. MobileHCI '11: *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*. ACM.

[14] Forgue, M., and Hazael-Massieux, D. 2012. Mobile Web Applications: Bringing Mobile Apps and Web Together. *Proceedings of the 21st international conference companion on World Wide Web* (16-20 April). Lyon, France. DOI=10.1145/2187980.2188022.

[15] Goadrich, M. H., and Rogers, M. P. 2011. Smart Smartphone Development: iOS versus Android. SIGCSE '11: *Proceedings of the 42nd ACM technical symposium on Computer science education*. ACM.

[16] Grønli, T., Hansen, J., and Ghinea, G. 2011. A Cloud on the Horizon: The Challenge of Developing Applications for Android and iPhone. PETRA '11: *Proceedings of the 4th International Conference on Pervasive Technologies Related to Assistive Environments*. ACM.

[17] Grønli, T., Hansen, J., and Ghinea, G. 2010. Android vs Windows Mobile vs Java ME: A Comparative Study of Mobile Development Environments. PETRA '10: *Proceedings of the 3rd International Conference on Pervasive Technologies Related to Assistive Environments*. ACM.

[18] Häkkilä, J., and Mäntyjärvi, J. 2006. Developing Design Guidelines for Context-Aware Mobile Applications. ProceedingMobility '06: *Proceedings of the 3rd international conference on Mobile technology, applications and systems* (Article No. 24). ACM, New York. DOI=10.1145/1292331.1292358.

[19] Huy, N. P., and Thanh, D. 2012. Evaluation of Mobile App Paradigms. MoMM '12: *Proceedings of the 10th International Conference on Advances in Mobile Computing and Multimedia*. ACM, 25-30. DOI=10.1145/2428955.2428968.

[20] Jones, M., and Marsden, G. 2006. *Mobile Interaction Design*. John Wiley & Sons, Ltd.

[21] Leonard, R., and Ruby, S. 2007. *RESTful Web Service*, O'Reilly Media, ISBN 978-0-596-52926-0

[22] Market Share Statistics for Internet Technologies. http://www.netmarketshare.com/mobile-market-share [Accessed on 12-05-2014]

[23] Microsoft Developer Network WCF Service http://msdn.microsoft.com/en-us/library/bb907578.aspx [Accessed 13-10-2014]

[24] Padley , R. 2011. HTML5 - Bridging the Mobile Platform Gap: Mobile Technologies in Scholarly Communication. Serials: *The Journal for the Serials Community* (0953-0460).

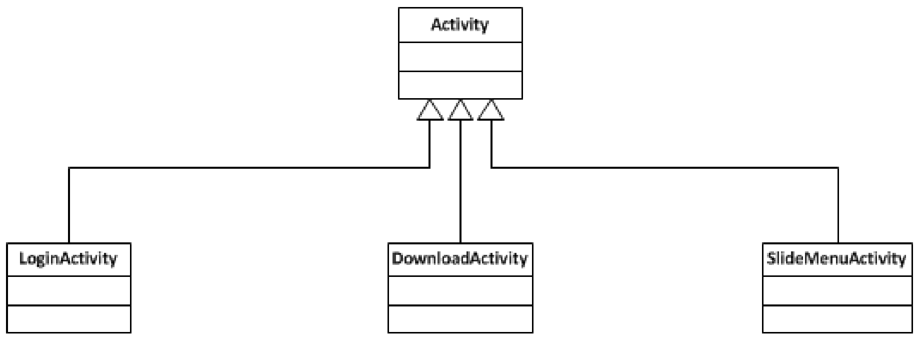
UKSG, Volume 24, Supplement 3, S32-S39. DOI=10.1629/24S32.

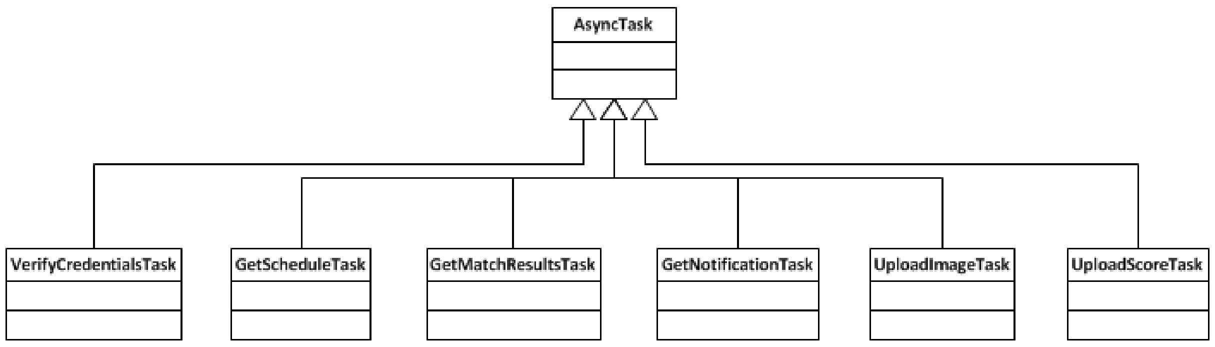
[25] Sletholt, M. T., Hannay, J., Pfahl, D., Benestad, H. C., and Langtangen, H.P. 2011. A Literature Review of Agile Practices and Their Effects in Scientific Software Development. SECSE '11: *Proceedings of the 4th International Workshop on Software Engineering for Computational Science and Engineering*. ACM.

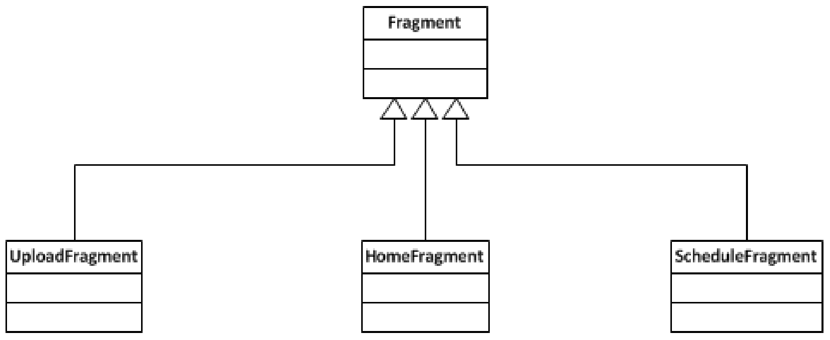
[26] Wasserman, A. I. 2010. Software Engineering Issues for Mobile Application Development. FoSER '10: *Proceedings of the FSE/SDP workshop on Future of software engineering research*. ACM.

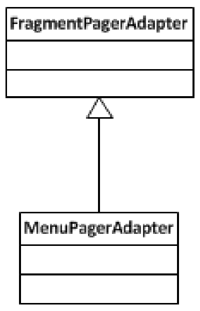
10. Appendices

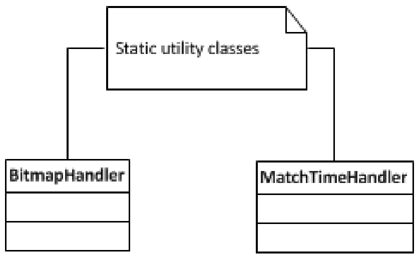
Appendix A: Android Class Hierarchy Diagram



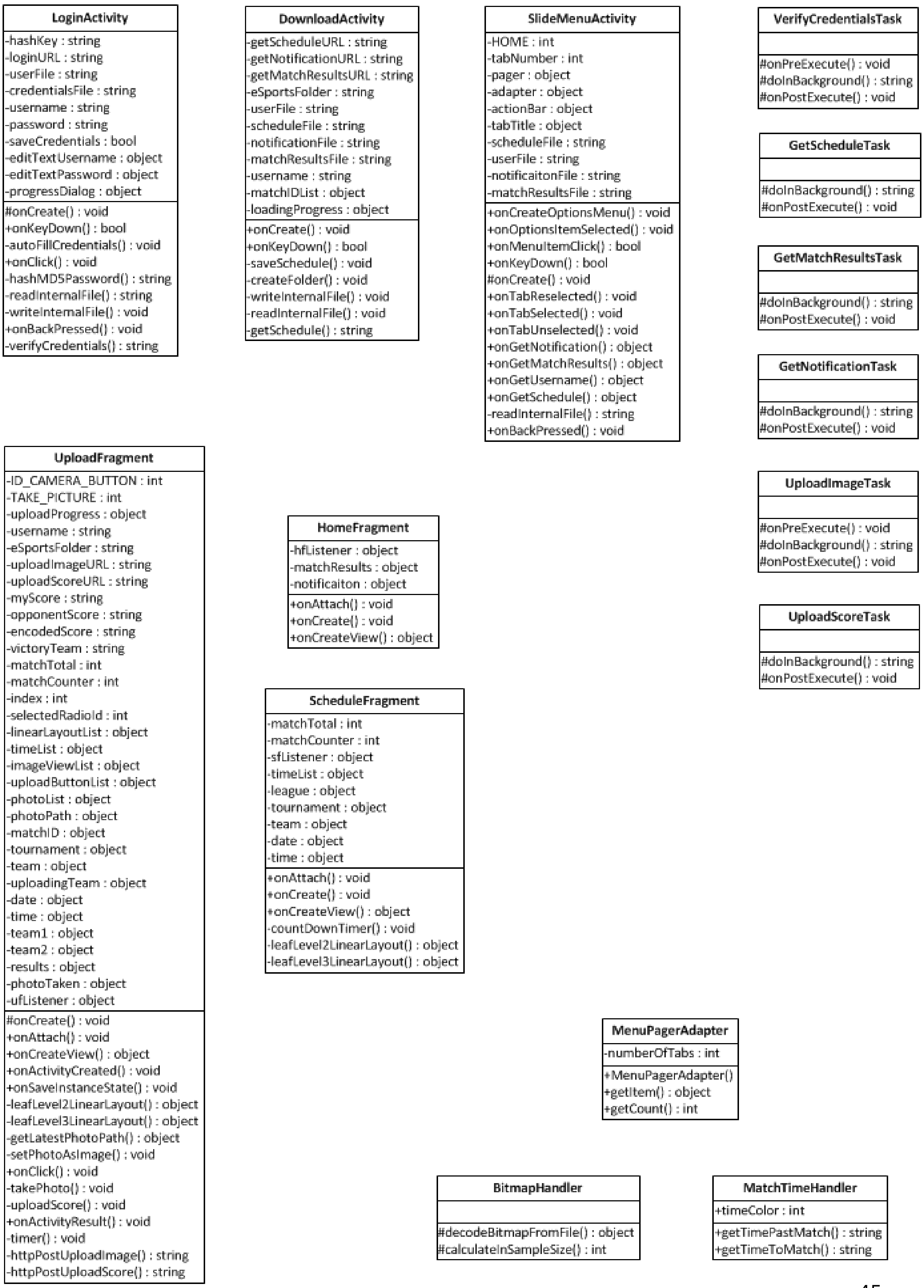




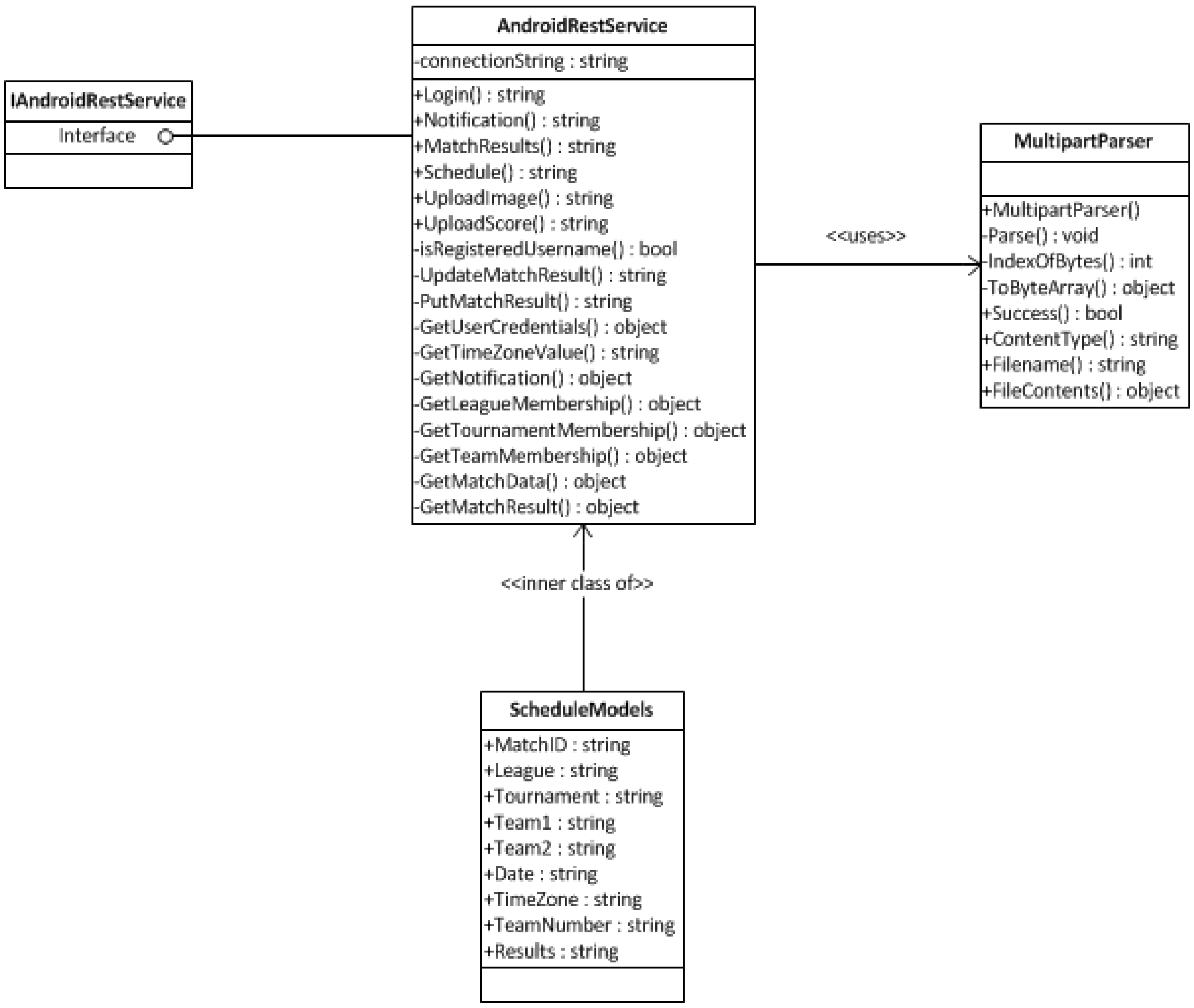




Appendix B: Android Class Description Diagram



Appendix C: Web-service Class Description Diagram



IAndroidRestService
Interface
AndroidRestService
-connectionString : string
+Login() : string
+Notification() : string
+MatchResults() : string
+Schedule() : string
+UploadImage() : string
+UploadScore() : string
-isRegisteredUsername() : bool
-UpdateMatchResult() : string
-PutMatchResult() : string
-GetUserCredentials() : object
-GetTimeZoneValue() : string
-GetNotification() : object
-GetLeagueMembership() : object
-GetTournamentMembership() : object
-GetTeamMembership() : object
-GetMatchData() : object
-GetMatchResult() : object
<<uses>>
MultipartParser
+MultipartParser()
-Parse() : void
-IndexOfBytes() : int
-ToByteArray() : object
+Success() : bool
+ContentType() : string
+Filename() : string
+FileContents() : object
<<inner class of>>
ScheduleModels
+MatchID : string
+League : string
+Tournament : string
+Team1 : string
+Team2 : string
+Date : string
+TimeZone : string
+TeamNumber : string
+Results : string