|  |
| --- |
|  |
| Fotoball |
|  |
| CSC509: Project Document |

April 23, 2015

Authored by: David Cipoletta, AbrAhAm Herrera, Adam Jilling, Rick Rejeleene

Rev 1.0

Fotoball

CSC509: Project Document

Contents

[Chapter 1: Requirements 3](#_Toc417567990)

[1.1 Introduction 3](#_Toc417567991)

[1.2 Glossary 4](#_Toc417567992)

[1.3 System Models 5](#_Toc417567993)

[1.4 Constraints 7](#_Toc417567994)

[Time: 7](#_Toc417567995)

[Hardware: 7](#_Toc417567996)

[Software: 7](#_Toc417567997)

[1.5 System Evolution 8](#_Toc417567998)

[1.6 Requirements Specification 9](#_Toc417567999)

[Hardware: 9](#_Toc417568000)

[Security and Safety Requirements 9](#_Toc417568001)

[Database Requirements 9](#_Toc417568002)

[1.7 Functional Requirements Definition 10](#_Toc417568003)

[1.8 Management Issues 11](#_Toc417568004)

[Schedule 11](#_Toc417568005)

[Technical Skills 11](#_Toc417568006)

[1.9 Disasters 12](#_Toc417568007)

[Chapter 2: Conceptual Models 13](#_Toc417568008)

[2.1 Overview Model 13](#_Toc417568009)

[2.2 Hardware Conceptual Model 14](#_Toc417568010)

[2.3 Software Conceptual Model 15](#_Toc417568011)

[2.4 Legend 16](#_Toc417568012)

[Chapter 3: SEQUENCE DIAGRAM 17](#_Toc417568013)

[3.1 Sequence Flow Diagram 17](#_Toc417568014)

[3.2 Database Diagram 18](#_Toc417568015)

[3.3 Hardware Sequence 19](#_Toc417568016)

[3.4 Use Case 21](#_Toc417568017)

[Chapter 4: CLASS MODELS 23](#_Toc417568018)

[4.1 Main Class Description 23](#_Toc417568019)

[4.2 AddNew Class Description 25](#_Toc417568020)

[4.3 LiveFeed Class Description 26](#_Toc417568021)

[4.4 About Class Description 27](#_Toc417568022)

[Chapter 5: TESTING & INTEGRATION PLAN 28](#_Toc417568023)

[5.1 Introduction 28](#_Toc417568024)

[5.1.1 Purpose 28](#_Toc417568025)

[5.1.2 Scope 28](#_Toc417568026)

[5.2 Modular 28](#_Toc417568027)

[5.2.1 Validation Requirements 28](#_Toc417568028)

[5.2.2 User Interface Requirements 29](#_Toc417568029)

[5.2.3 Performance Requirements 30](#_Toc417568030)

[5.2.4 User Platform Requirements 31](#_Toc417568031)

[5.2.5 Fotoball Device Access requirements 32](#_Toc417568032)

[5.2.6 Hardware Requirements 33](#_Toc417568033)

[5.2.7 Database Requirements 34](#_Toc417568034)

[5.3 System Integration 35](#_Toc417568035)

[5.3.1 Integration Phase -1 - Testing system components independently 35](#_Toc417568036)

[5.3.2 Integration Phase -2 – Database and Communication 35](#_Toc417568037)

[5.3.3 Integration Phase -3 Functionality and Performance 35](#_Toc417568038)

[5.3.4 Integration Phase -4 –Complete system Integration software and hardware 36](#_Toc417568039)

[5.4 Testing Schedule 36](#_Toc417568040)

[Chapter 6: PROBLEMS ENCOUNTERED 38](#_Toc417568041)

[Chapter 7: FOTOBALL PROJECT FINAL SUMMARY 40](#_Toc417568042)

# Chapter 1: Requirements

## 1.1 Introduction

The purpose of this document is to specify the hardware and software requirements of the Fotoball project. The project was derived with the idea that camera technology could be much further woven into the fabric of sports, specifically football, than it currently is.

The most recent camera upgrades to the way games are broadcast involve “skycams” where a camera is suspended by a series of cables above the field and remotely controlled to provide a bird’s eye view of the action. There have also been brief experiments involving “helmet cams” where a small camera is mounted to a player’s helmet to provide first-hand game action. After some research, it was realized that very little has been done involving attaching cameras directly to the football. The only previous attempt of note was a side-mounted camera that takes a still image after each rotation and ultimately pieces together a choppy video that is oriented in the wrong direction.

Our project will involve mounting two cameras to each end of the football to be able to capture a front and rear view of where the football is heading and where it came from. The data captured by each camera will then be wirelessly streamed to a user’s mobile device, most likely a smart phone. From there the video can be saved, edited, and shared however the user chooses.

There are two potential markets of interest for the product. The first is as a recreational device for kids and young adults to be able to film fun videos and share with their friends. Much like the GoPro has revolutionized extreme sports like skiing, surfing, and rock climbing, we feel this product could do the same for football. The second market of interest is in high-level televised NFL and college games. This is more of a challenge since the ball would have to adhere to extremely specific guidelines, but the benefits the fan watching on television could be huge.

## 1.2 Glossary

Alpha testing - a simulated or actual operational testing by potential users/customers or an independent test team at the developers' site.

Android - a mobile operating system developed by Google and run on many different devices

C++ - a general-purpose programming language.

GoPro - a compact, often wearable, camera that is popular in the use of filming in extreme situations that would render a traditional camera unfit

iOS - Apple’s mobile operating system run on all Apple mobile devices

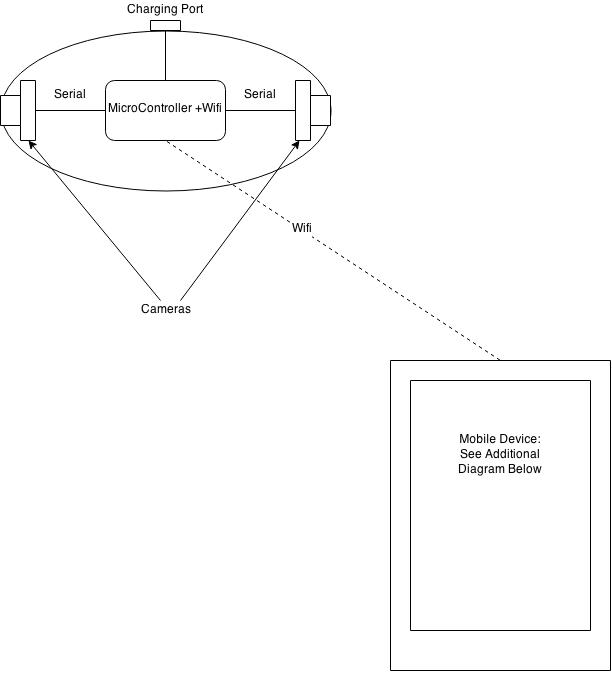
Java - an object-oriented programming language that is maintained by Oracle Corporation

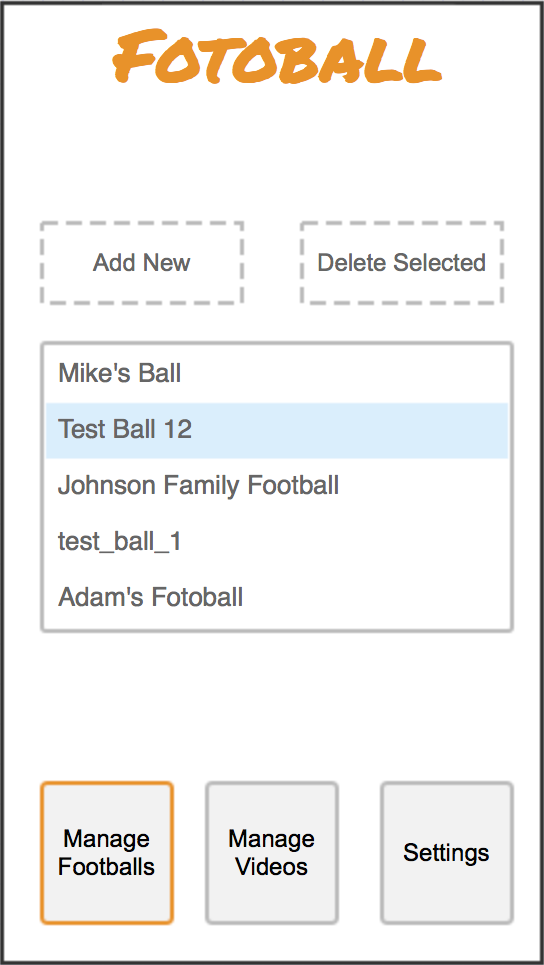
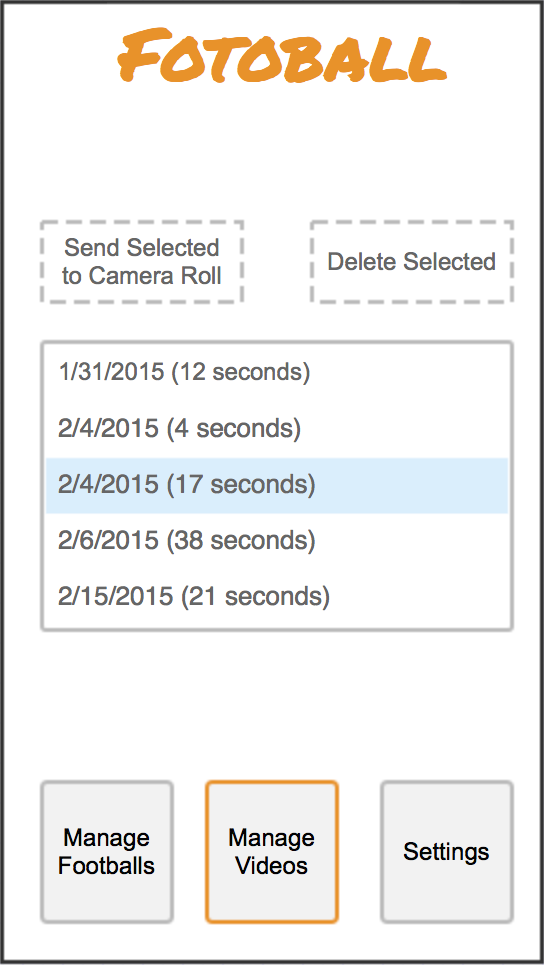
Microcontroller - a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals.

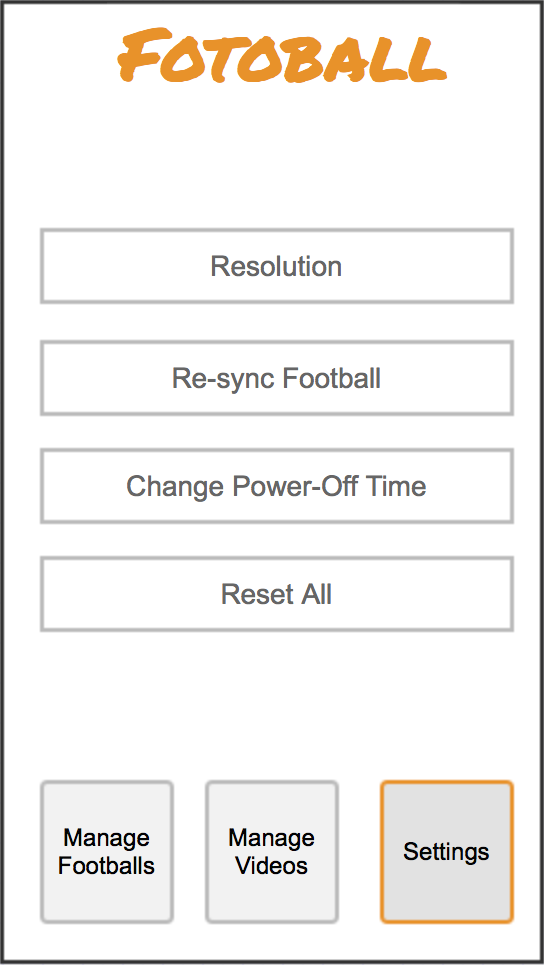
Resolution - the degree of sharpness of a computer-generated image as measured by the number of dots per linear inch in a hard-copy printout or the number of pixels across and down on a display screen.

Serial - the process of sending data one bit at a time, sequentially, over a communication channel or computer bus.

## 1.3 System Models





## 1.4 Constraints

### Time:

Product and code design will be completed by 02/19/2015

Prototype completion will be completed by 03/13/2015

Alpha testing will be completed by 03/26/2015

Testing model will be completed by 04/09/2015

Final test will be completed by 04/16/2015

Final presentation will be completed by 04/23/2015

### Hardware:

                            The microcontroller needs to weigh under 25g.

                            The microcontroller needs to communicate via Wi-Fi.

                            The microcontroller needs to store video locally.

                            The camera needs to weigh under 20g.

The electronics need to be powered for at least 4 hours.

### Software:

                            The mobile application will be programmed in Java.

                            The firmware will be programmed in C/C++.

## 1.5 System Evolution

There are several critical functions that the first iteration of our system must have.

1. The hardware must be attached to the ball in a way that keeps the ball as close to its originally properties as possible. Weight needs to be consistent, the overall shape needs to remain the same, and the ball must remain balanced. There is a certain amount of leeway with which to work, but not a lot. If the ball does not behave and act like a regular football, the rest will be meaningless.
2. The video data must reliably transmit to the user’s mobile device. If a throw is not recorded and becomes “lost” the user will get frustrated and likely stray away from the product. Internal storage is an option currently being considered as an attempt to act as a backup should this happen. Internal storage will also be an option to act as a backup should this happen.
3. The image must be stabilized enough to make the video an acceptable quality. This is perhaps our biggest challenge, and likely why this product does not currently exist. Our system needs to compensate for every movement of the football as it rotates and wobbles, and ultimately return a smooth video.
4. The software will have to be compatible with both Apple and Android devices. To not support one of these platforms would be ignoring too big of a market segment for the endeavor to be worthwhile.

As the system evolves, there are additional features that would fit in nicely. One idea is to add additional sensors capable of measuring the speed, location, rotation, and orientation of the ball. This data could then be extrapolated and used to provide “throw quality” feedback that will tell user how far the ball went, the ball’s velocity, and how tight of a spiral it was.

Another goal is to gradually improve the quality of components to provide higher resolution video and even audio.

## 1.6 Requirements Specification

### Hardware:

            The electronics shall be lightweight.

            The electronics shall not affect footballs throw.

            The electronics shall be able to transmit data wirelessly.

### Security and Safety Requirements

                            The electronics shall not be accessible by any unauthorized devices.

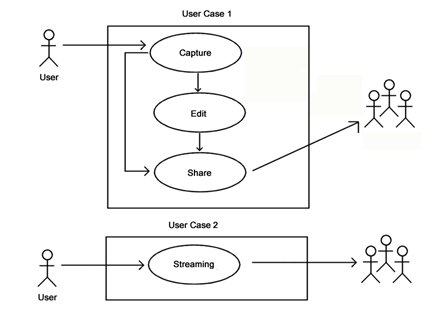
### Database Requirements

                            The software shall be able to save video locally.

## 1.7 Functional Requirements Definition

The Fotoball device will be used mainly with these two objectives in mind:

* Recreational purposes. - User will be able to capture, edit, and share still images or videos from their mobile devices.
* Live streaming. - The two cameras attached to the Fotoball device will make the user feel like he/she is part of the action.



Additional features could be added for training drills purposes, such as gathering information (throwing and rotational speed, tossed angle, etc.), which will help the user to improve his/her "throw quality."

## 1.8 Management Issues

### Schedule

|  |  |
| --- | --- |
| **Date** | **Process** |
| February 19 | Product + Coding Design |
| March 13 | Prototype Completion |
| March 26 | Alpha Testing |
| ***April 09*** | ***Testing Model (in class)*** |
| April 16 | Final Test |
| ***April 23*** | ***Presentation of Final Project*** |

## Technical Skills

* Hardware expertise
* Coding skills
* App development

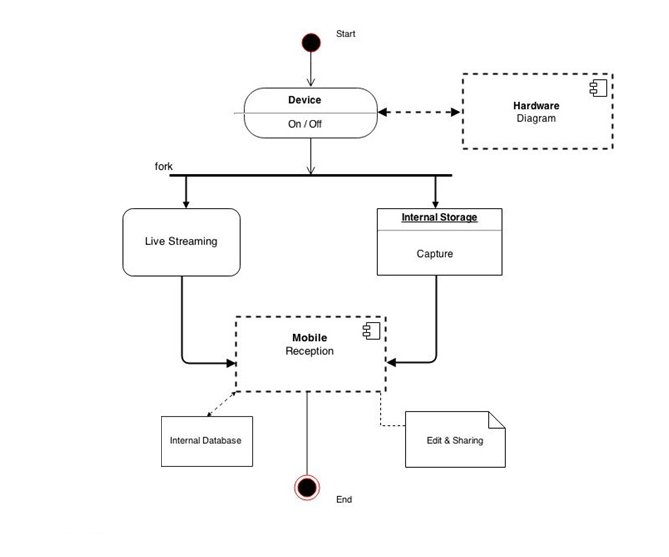
## 1.9 Disasters

* If we must face any of those circumstances, a backup plan has been elaborated to overcome these issues. We are considering the following scenarios and how we plan to deal with them:
  + Not getting all the required hardware on time to construct a high quality product: we have proposed a schedule which will allow us enough time to work around any issue.
  + The image/video streaming to the user's mobile device gets lost or interrupted: an internal storage option will be added as a backup solution.
  + The image/video streaming gets too blurred during the action: a real-time digital image stabilization technique will be used to counteract the motion.

# Chapter 2: Conceptual Models

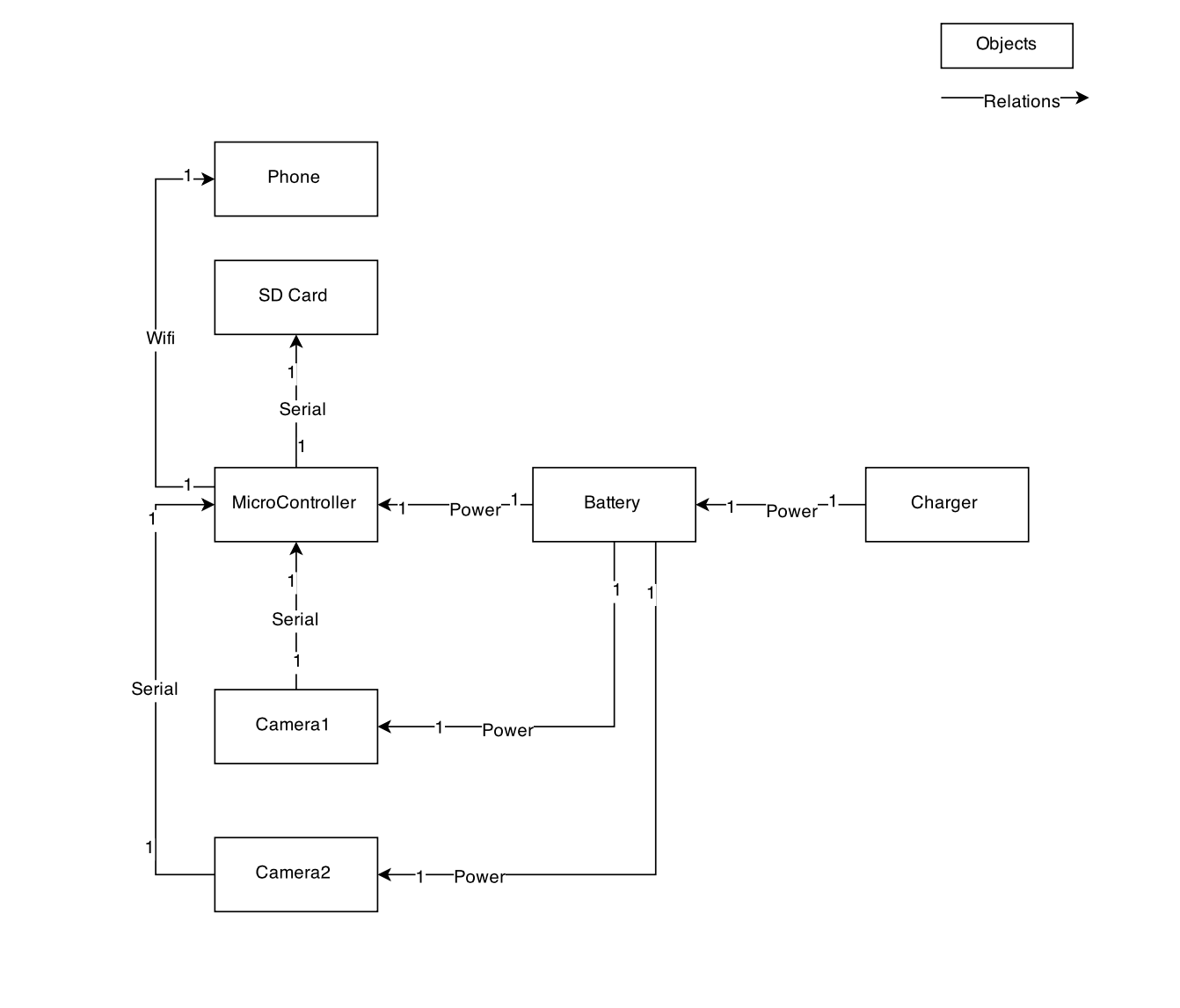
## 2.1 Overview Model

The following represents the overall flow of the system. From the Fotoball hardware, to the users mobile device, to the internal database.



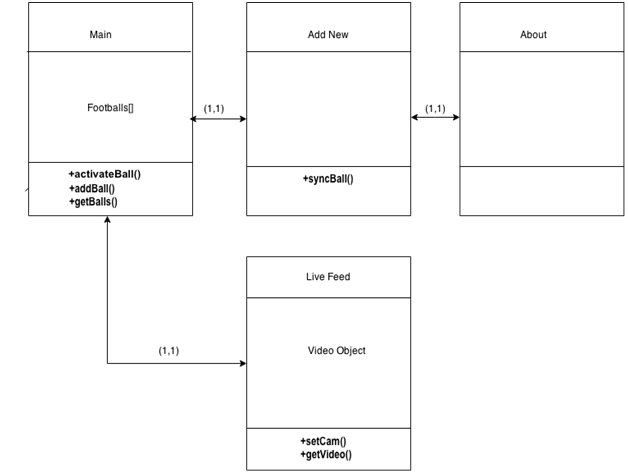
## 2.2 Hardware Conceptual Model

Basic flow of the hardware system showing which components will connect to which other components.

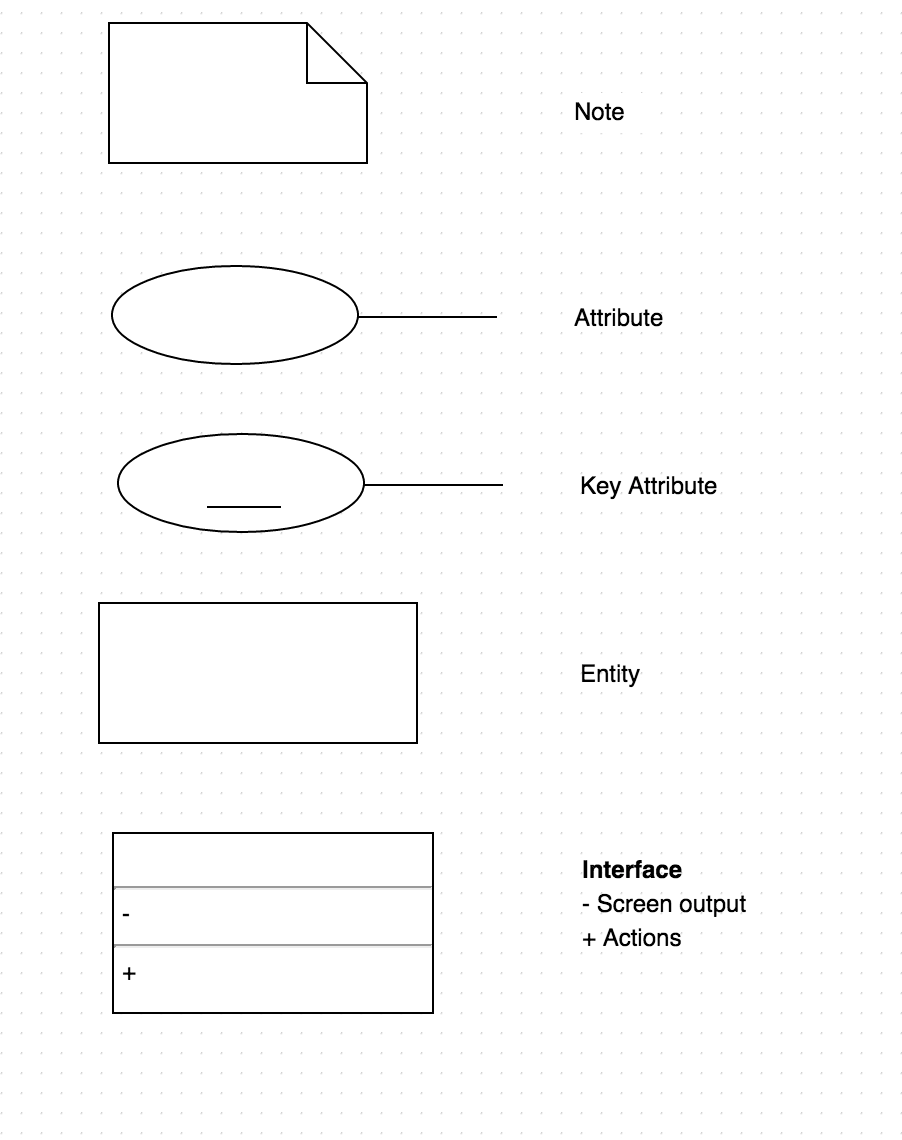


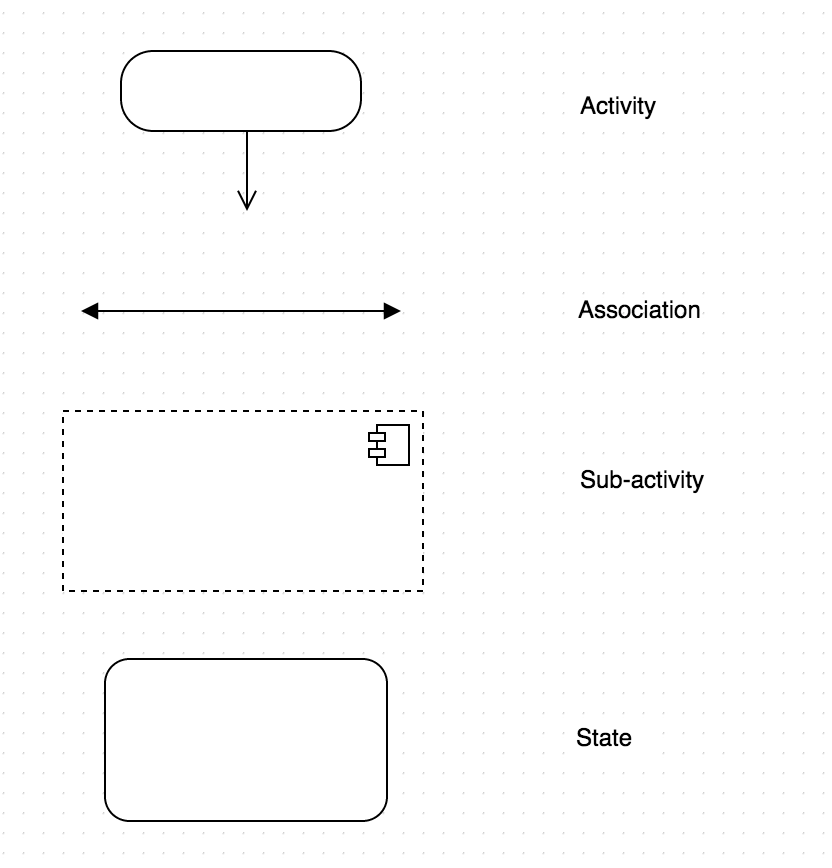
## 2.3 Software Conceptual Model

The main classes and views that make up the application are shown here, with each class’s objects represented in the middle and each class’s major methods represented in the bottom.



## 2.4 Legend

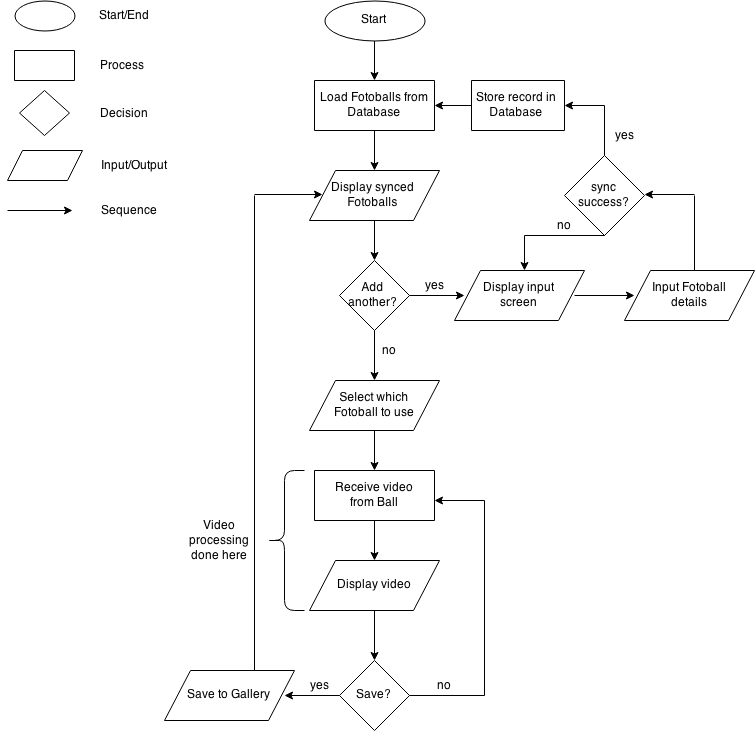




# Chapter 3: SEQUENCE DIAGRAM

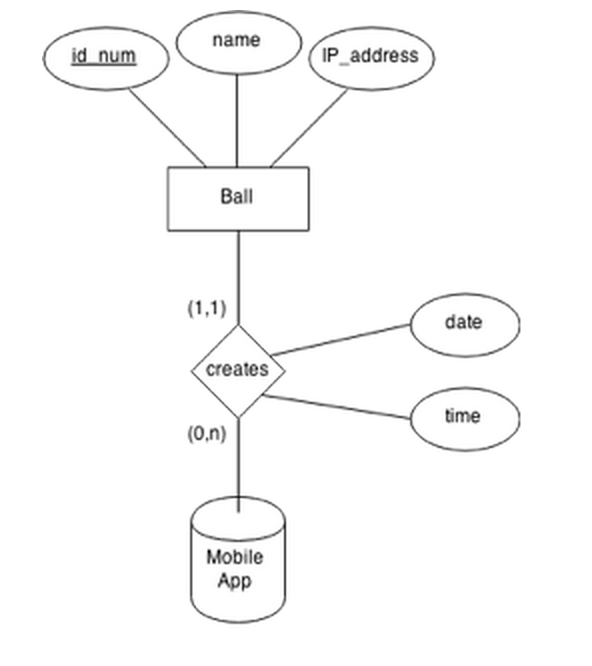
## 3.1 Sequence Flow Diagram

The following chart represents the flow of control through the software system. All videos will be saved to the internal media gallery of the parent device.



## 3.2 Database Diagram

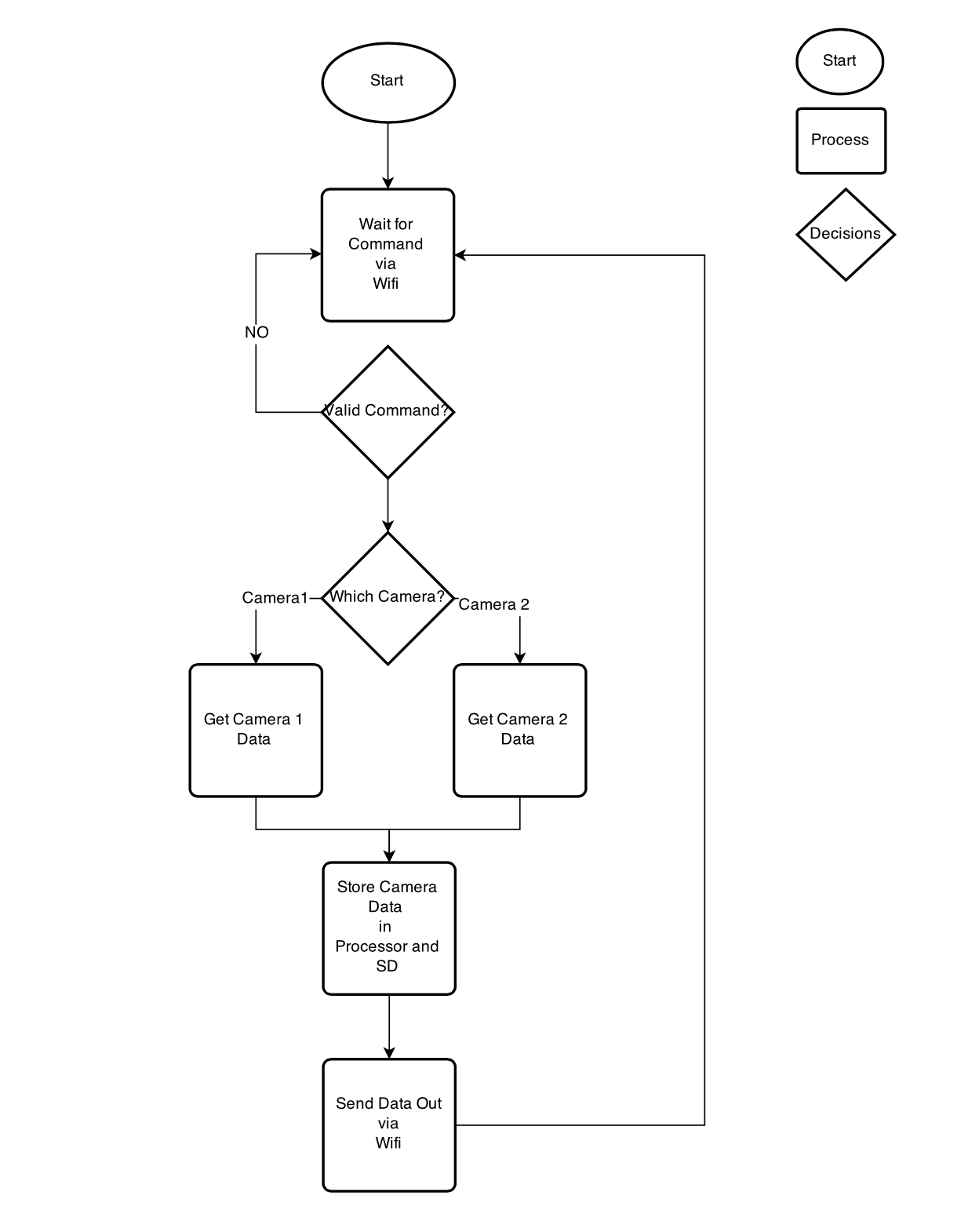
The database will connect the ball information with the mobile application. Anywhere from zero to n balls will be allowed to be created and the time and date of each creation will stored, as well as the relevant ball information.

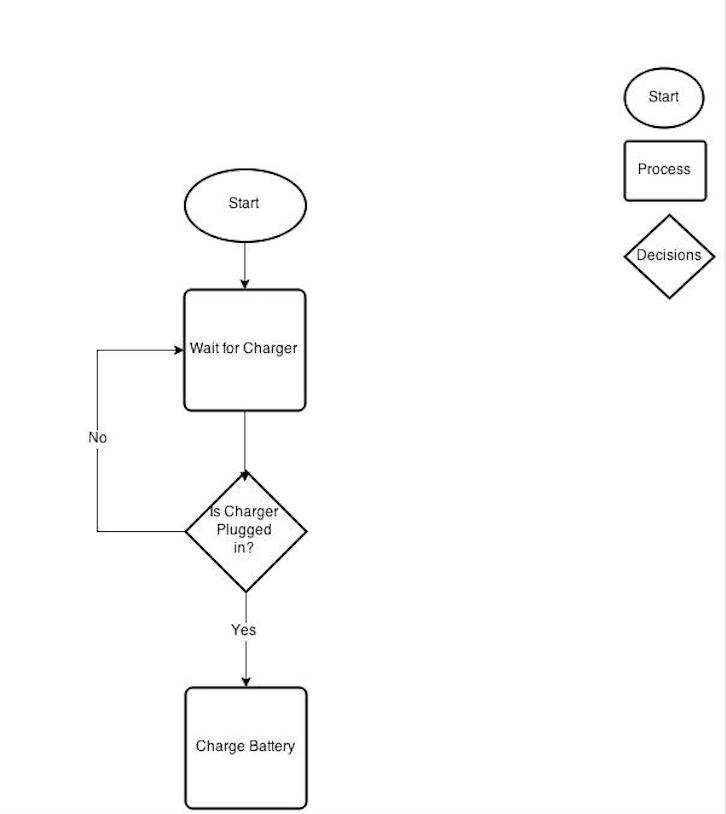


### 

## 3.3 Hardware Sequence

This represents the sequence of flow from the hardware system’s perspective. Cameras will be switched manually via hardware.

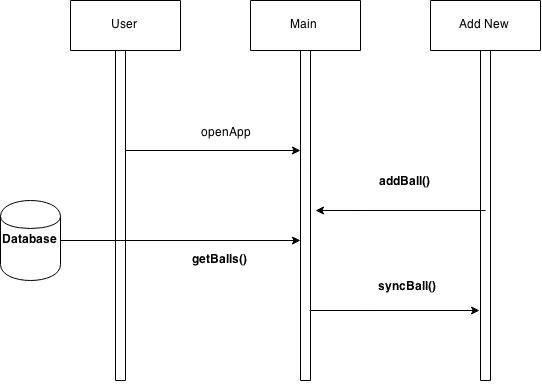




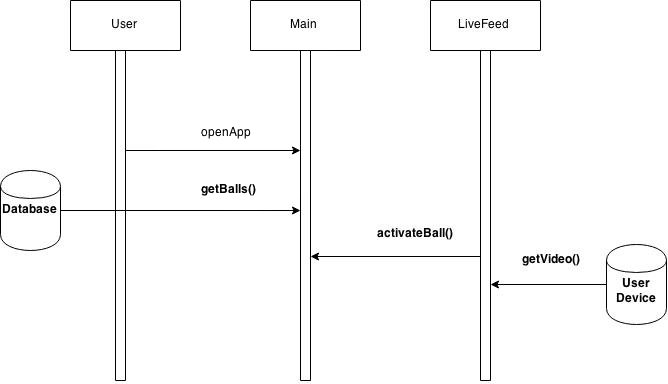
## 3.4 Use Case

This App shows the user interaction with the user and the football

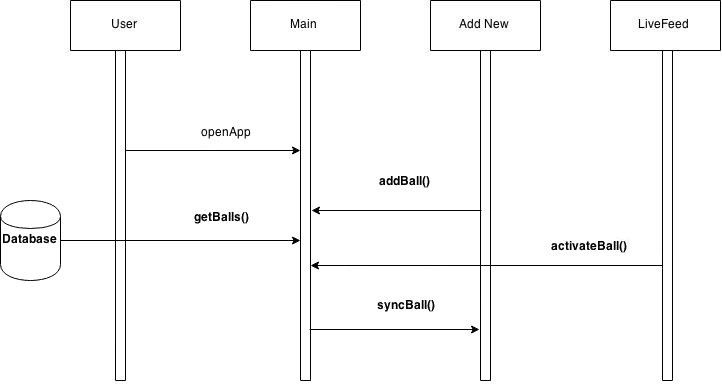
Use Case 1: User checking a new Fotoball



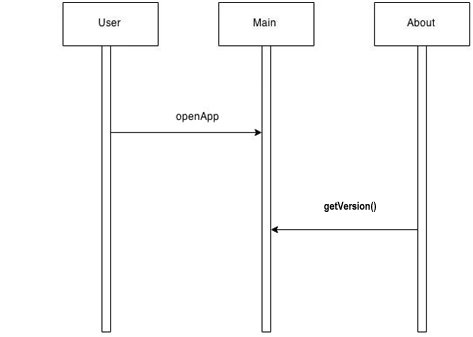
Use Case 2: User checking a live stream from the list of sync'ed Fotoball streams and save it on his personal device



Use Case 3: User adding a new Fotoball and selecting a live stream



Use Case 4: User checking the About information



# Chapter 4: CLASS MODELS

## 4.1 Main Class Description[[1]](#footnote-1)

|  |  |
| --- | --- |
| **Name** | Main |
| **Base Class** | <none> |
| **Purpose** | Display the list of synced Fotoball streams and provide the option to select a ball or load a new streams |
| **States** | Active, Inactive |
| **Constructors** | Default: empty list  Fotoball array: load existing Fotoballs |
| **Mutators** | activateBall()  addBall() |
| **Accessors** | getBalls() |
| **Fields** | Fotoballs[] |

**activateBall() Method Description**

|  |  |
| --- | --- |
| **Prototype** | private void activateBall(Object) |
| **Purpose** | Connects to a given Fotoball stream |
| **Receives** | a Fotoball object |
| **Returns** | nothing |
| **Remarks** | Will throw an exception if unsuccessful |

**addBall() Method Description**

|  |  |
| --- | --- |
| **Prototype** | private void addBall() |
| **Purpose** | Connects to the AddNew view |
| **Receives** | nothing |
| **Returns** | nothing |
| **Remarks** | This method will link to the AddNew class |

**getBalls() Method Description**

|  |  |
| --- | --- |
| **Prototype** | private Object[] getBalls() |
| **Purpose** | Queries the Fotoball database and displays the list of synced Fotoball streams (if any) |
| **Receives** | nothing |
| **Returns** | array of Football objects |
| **Remarks** | Will display a message if no Fotoballs are found |

## 4.2 AddNew Class Description[[2]](#footnote-2)

|  |  |
| --- | --- |
| **Name** | AddNew |
| **Base Class** | <none> |
| **Purpose** | Provides an interface to connect a new Fotoball |
| **States** | Active, Inactive |
| **Constructors** | Default only |
| **Mutators** | syncBall() |
| **Accessors\*** | getName()  getIP()  getPort() |
| **Fields** | Name: string  IP: long integer  Port: integer |

**syncBall() Method Description**

|  |  |
| --- | --- |
| **Prototype** | private void syncBall(String name, long ip, int port) |
| **Purpose** | Uses inputted data to connect with an external Fotoball |
| **Receives** | name, IP address, port number |
| **Returns** | nothing |
| **Remarks** | Will throw an exception if unsuccessful |

\*No description for three getters since they are fairly self-explanatory

### 

## 4.3 LiveFeed Class Description[[3]](#footnote-3)

|  |  |
| --- | --- |
| **Name** | LiveFeed |
| **Base Class** | <none> |
| **Purpose** | LiveFeed displays the video image being transmitted by the Fotoball and provides user options to change views or to capture video data |
| **States** | Active, Inactive |
| **Constructors** | Default: no camera loaded  Ball Object: sends one Fotoball object to be used |
| **Mutators** | setCamera() |
| **Accessors** | getVideo() |
| **Fields** | video\_feed |

**setCamera() Method Description**

|  |  |
| --- | --- |
| **Prototype** | private void setCamera(int camNum) |
| **Purpose** | Specify which camera to use |
| **Receives** | camNum - an integer specifying which camera to turn on |
| **Returns** | nothing |
| **Remarks** | This method will also make sure every other camera is turned off, so only one is live at a time |

**getVideo() method Description**

|  |  |
| --- | --- |
| **Prototype** | public Object getVideo() |
| **Purpose** | To extract the video in use and save it to the device gallery |
| **Receives** | nothing |
| **Returns** | The current video object from the LiveFeed screen |
| **Remarks** | Will throw an exception if the video is unable to be returned |

## 4.4 About Class Description[[4]](#footnote-4)

|  |  |
| --- | --- |
| **Name** | About |
| **Base Class** | <none> |
| **Purpose** | Display creator information, version details, and link to website |
| **States** | Active, Inactive |
| **Constructors** | Default only |
| **Mutators** | none |
| **Accessors** | getVersion() |
| **Fields** | version: float |

# Chapter 5: TESTING & INTEGRATION PLAN

## 5.1 Introduction

5.1.1 Purpose - The main purpose of the system integration and testing is to validate the hardware and software.

This is to make sure the product released is ready to be used by the public. The Integration phase includes test cases for each module, as well as testing to ensure that every module ties together properly.

5.1.2 Scope - The integration testing routines follow the "White Box" approach to test and integrate the software. The white box approach will help us to understand the behavior of each module and to build quality software. Integration is carried on a modular approach.

## 5.2 Modular

### 5.2.1 Validation Requirements

#### 5.2.1.1 Validation Requirements

|  |  |
| --- | --- |
| Requirement Number | 2.1 |
| Requirement Name | Validation Requirements |
| Requirement Description | Validation is handled via the mobile device’s internal security system |
| Pre-conditions |  |
| Procedures | Since our system will not verify individual users, just individual balls, validation can be piggy-backed to the mobile system’s verification. The app will live within the device’s existing ecosystem, thus transferring any user clearances to the device’s operating system.  If a user is provisioned to use the device, they are provisioned to use the application. |
| Post Conditions | The system will return the message: user the validation is completed. |
| Test Results | PASS |

### 5.2.2 User Interface Requirements

#### 5.2.2.1- User Interface - GUI

|  |  |
| --- | --- |
| Requirement Number | 2.21 |
| Requirement Name | User Interface – GUI for Fotoball |
| Requirement Description | All users can access the GUI for Fotoball |
| Pre-conditions | The user must have the application installed on the phone. |
| Procedures | 1. Using your mobile device, locate the Fotoball application 2. Click on the application 3. The user will be see the Main Fotoball screen. |
| Post Conditions | The Fotoball GUI can be controlled by the user |
| Test Results | PASS |

#### 5.2.2.2 User Interface - To Allow authorized individuals to view the Fotoball

|  |  |
| --- | --- |
| Requirement Number | 2.22 |
| Requirement Name | User Interface |
| Requirement Description | Allow authorizing individuals to live stream from Fotoball |
| Pre-conditions | The user must be logged into the system. |
| Procedures | 1. On the Main screen, click activate ball. 2. Now the user will be asked for the required Fotoball details 3. Enter the details and proceed |
| Post Conditions | The system will display a message that confirms the Fotoball access and live stream |
| Test Results | PASS |

### 5.2.3 Performance Requirements

#### 5.2.3.1 - Performance Requirements - Booting up/ live stream authentication

|  |  |
| --- | --- |
| Requirement Number | 2.31 |
| Requirement Name | Performance Requirements – Booting up and Live stream authentication |
| Requirement Description | Users will be able to boot into the hardware through Fotoball application and access the live stream. |
| Pre-conditions | User must have the required access details for Fotoball.  Measuring it using a chronograph |
| Procedures | 1. Open the application 2. Click on the activate ball 3. Enter the required Fotoball authentication 4. Click on submit   (<5 seconds) |
| Post Conditions | The system will verify the details and show the live stream from Fotoball device |
| Test Results | PASS |

#### 5.2.3.2- Performance Requirements – Accessing video and live streaming it back to the application

|  |  |
| --- | --- |
| Requirement Number | 2.32 |
| Requirement Name | Performance Requirements – Accessing video and live streaming it back to the application. |
| Requirement Description | The User will be able to access the video and live stream it from the Fotoball hardware into the application. |
| Pre-conditions | User must have the Fotoball hardware and authentication requirements for accessing live stream.  Have a time measurements device available. (chronometer) |
| Procedures | 1. Open the application in mobile device 2. Click on the activate ball option 3. Using your chronometer, be ready for calculating the performance 4. Enter the required Fotoball authentication and click submit 5. Start the chronometer simultaneously when you click submit 6. Measure the time taken from accessing the live stream to the application   (It should be less than 5 seconds) |
| Post Conditions | The System will be in live feed window |
| Test Results | PASS |

### 5.2.4 User Platform Requirements

#### 5.2.4.1 Test User Platform Requirements

|  |  |
| --- | --- |
| Requirement Number | 2.41 |
| Requirement Name | User Platform Requirements |
| Requirement Description | The User platform is to be initially tested in a desktop environment using an Android-like simulator |
| Pre-conditions | The User needs a Device with an Android Operating System |
| Procedures | 1. Find an Android device or simulator 2. Verify that it is running Android version 5.0 or later 3. Connect the Device to the System. 4. Install the application initially from an external source that is provided to the user |
| Post Conditions | * If the Application is successfully installed, it will open. * If the Application is not successfully installed, it will prompt the user “Application installation failed” |
| Test Results | PASS |

### 5.2.5 Fotoball Device Access requirements

#### 5.2.5.1 Test Fotoball Device Access requirements- full name

|  |  |
| --- | --- |
| Requirement Number | 2.51 |
| Requirement Name | Fotoball Device Access requirements |
| Requirement Description | The Fotoball device will contain specific name and IP address to access it. The user will need to know the details of the Fotoball to access it |
| Pre conditions | The User needs to be in Main page |
| Procedures | 1. Enter the Name 2. Enter the IP Address. 3. Enter the valid information (network, port number) 4. Click Submit after entering the above information. |
| Post Conditions | The User will access the Fotoball live stream if all the information matches it. |
| Test Results | PASS |

#### 5.2.5.2 Test User Account Requirements - Registered Fotoball

|  |  |
| --- | --- |
| Requirement Number | 2.52 |
| Requirement Name | User Account Requirements – Registered Fotoball |
| Requirement Description | The User account contains registered Fotoball device. |
| Pre conditions | The User needs to be in the Main menu and add a Fotoball device to their account |
| Procedures | * The User needs to fill valid information for the rest of the form to register the Fotoball device |
| Post Conditions | * The Default User is assigned to a Fotoball device |
| Test Results | Fail. Manual input not yet functional. |

### 5.2.6 Hardware Requirements

#### 5.2.6.1 Power Test

|  |  |
| --- | --- |
| Requirement Number | 2.61 |
| Requirement Name | Power Test |
| Requirement Description | Verify that Fotoball is able to be charged and retain charge |
| Pre-conditions | Battery < 50% charged |
| Procedures | 1. Power on Fotoball  2. Open the Fotoball  3. Check if the power LED lit  4. Use volt meter measure across +5V and Gnd and check the power for camera  5. Plug in Fotoball to a charger  6. Check battery voltage  7. Wait for 2 minutes  8. Check if battery voltage went up. |
| Post Conditions | Battery > 50% charge |
| Test Results | PASS |

#### 5.2.6.2 Communication Test

|  |  |
| --- | --- |
| Requirement Number | 2.62 |
| Requirement Name | Communication Test |
| Requirement Description | Ensure Fotoball is able to wirelessly sync with mobile device |
| Pre-conditions | Fotoball not connected |
| Procedures | 1. Power on Fotoball  2. Wait for 2 minutes  3. Check if you can see the Fotoball wireless SSID  4. Connect to Fotoball wireless SSID  5. Ping Fotoball IP address. |
| Post Conditions | Fotoball connected |
| Test Results | PASS |

### 5.2.7 Database Requirements

#### 5.2.7.1 Type Check Test

|  |  |
| --- | --- |
| Requirement Number | 2.71 |
| Requirement Name | Type-Check Test |
| Requirement Description | Ensure Fotoball database is immune to invalid input |
| Pre-conditions |  |
| Procedures | 1. Connect to Fotoball database via backend 2. Attempt to input erroneous data types for each relation of each table |
| Post Conditions | No invalid inputs allowed |
| Test Results | PASS |

#### 5.2.7.2 Concurrency/Stability Test

|  |  |
| --- | --- |
| Requirement Number | 2.72 |
| Requirement Name | Concurrency and Stability Test |
| Requirement Description | Ensure each Fotoball element can only be accessed by one external source at a time |
| Pre-conditions | At least 1 Fotoball loaded into database |
| Procedures | 1. Connect to Fotoball database via backend 2. Attempt to access/change information from source A 3. Attempt same access from source B |
| Post Conditions | Source B access is denied |
| Test Results | PASS |

## 5.3 System Integration

### 5.3.1 Integration Phase -1 - Testing system components independently

The first step of system integration plan will consist of making sure each one of the system components are working as given in the requirement specification. This consists of making sure the hardware and software are thoroughly testing and integrated. In the Fotoball Application, the main system components is fully operational and tested independently before trying to integrate or try to test any interaction with any of the other mayor system components. The phase-1 will validate the system have all the necessary components to start with a formal system integration. The integration phase will include, testing the application, hardware and the database independently

**Phase 1 Integration Testing:**

* Testing the application installation
* Testing GUI and classes that do not have dependencies
* Testing the hardware to make sure it is working
* Testing database installation and finding tables are created.

### 5.3.2 Integration Phase -2 – Database and Communication

The Phase 2 of the system integration will consist of integrating the database system with the application. As part of this phase, we will test the communications interfaces. In addition, this phase includes testing the functionality and the interaction between the server and the application

**Phase 2 Integration Testing:**

* Testing Wireless communication between the application and hardware
* Testing Server

### 5.3.3 Integration Phase -3 Functionality and Performance

The Phase 3 of the system integration will consist of testing the functionality of integrated modules and performance. As a part of this test, we will conduct the performance test of the live stream from the hardware to the system GUI.

**Phase 3 Integration Test**

* Testing Functionality
* Testing Performance

### 5.3.4 Integration Phase -4 –Complete system Integration software and hardware

The Phase 4 of the system integration will test and integrate all the system components as a one whole system. This will include testing both the hardware and software component together

**Phase 4 Integration Test**

* Testing Wireless between all systems simultaneously (Hardware/Software)
* Testing client to database request
* Testing database to Fotoball
* Testing system performance requirements.

## 5.4 Testing Schedule

|  |  |
| --- | --- |
| Test Schedule ID: | FbT-0001 |
| Product ID / Name: | Fotoball |
| Product Version: | v1.0 |
| Created On: | Document created on April 01, 2015 |
| Review On: | Document reviewed on April 08, 2015 |
| Reviewed By: | Cipoletta, David  Herrera, AbrAhAm Jilling, Adam Rejeleene, Rick |
| Current Status: | PASS |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Step** | **Start Date** | **End Date** | **Responsibility / Comments** |
| 1. Hardware Requirements | Jan 29, 2015 | Feb 19, 2015 | PASS |
| 2. Fotoball Device Access requirements | Feb 23, 2015 | Feb 26, 2015 | PASS |
| 3. User Interface | Mar 02, 2015 | Mar 05, 2015 | PASS |
| 3.1 GUI installation | Mar 02, 2015 | Mar 05, 2015 | PASS |
| 3.2 Login access | Mar 02, 2015 | Mar 05, 2015 | Not yet implemented |
| 4. Database Requirements | Mar 09, 2015 | Mar 12, 2015 | PASS |
| 5. Performance Requirements | Mar 23, 2015 | Mar 26, 2015 | PASS |
| 5.1 Live Streaming Authentication Performance | Mar 23, 2015 | Mar 26, 2015 | PASS |
| 5.2 Live Streaming Performance | Mar 23, 2015 | Mar 26, 2015 | PASS |
| 6. System Integration | Mar 30, 2015 | Apr 02, 2015 | PASS |
| 6.1 Testing system components independently | Mar 30, 2015 | Apr 02, 2015 | PASS |
| 6.2 Database and Communication | Mar 30, 2015 | Apr 02, 2015 | PASS |
| 6.3 Functionality and Performance | Mar 30, 2015 | Apr 02, 2015 | PASS |
| 6.4 Complete system Integration software and hardware | Mar 30, 2015 | Apr 02, 2015 | Partial PASS |
| Review / Approve the Final Test Report | Apr 13, 2015 | Apr 16, 2015 | Final test will be performed a week prior the last day of class |
| **Test Step** | **Start Date** | **End Date** | **Responsibility / Comments** |

# Chapter 6: PROBLEMS ENCOUNTERED

The Project went very well as the original specifications but due to time constraints, we were unable to develop the full product.

There were no major changes from the original specification till the proposed prototype.

Below is a summary of the encountered challenges for this project.

Hardware:

Although the capability exists to transmit video via Wi-Fi there are certainly many problems that arose that were not anticipated.

Power Consumption:

After extensive research on microcontrollers with Wi-Fi capabilities, we found that low power Wi-Fi microcontrollers are used to control simple I/O and are not meant to handle complex image processing.

This led us to using a microcontroller with SoC (system on a chip). These microcontrollers have relatively high power consumption since they have to process the image data, then encode it, and then stream it out.

Due to the high power consumption, the battery we have to use must be a high capacity and high voltage. This impacts the weight of Fotoball since the Fotoball will now be a bit heavier.

Latency:

Since we are using microcontrollers to do the image processing and encoding, there is a latency to the video broadcast. The latency is not significant but it is certainly something we did not plan on originally.

Signal Strength:

Since the ball will communicate via Wi-Fi, we have to use small antennae to fit into the ball. This causes the signal strength to drop slightly.

Time Constraints:

Certainly we did not explore some of the options such as streaming two cameras at once and implementing image stabilization. This is due to time and money constrains. One of the Software development kits for streaming video cost $3000. Additionally we ran out of time to implement the second camera.

libGDX:

The original plan was to use a development framework called libGDX to write one program and port it to both the iOS and Android platforms. Something with this capability sounds too good to be true and we soon learned that it was. Too many sacrifices needed to be made to be able to use libGDX and we decided it was in our best interest to write two separate projects for each. We started with the iOS version and coded it as most iOS are - in Xcode with Swift.

Apple webView:

Apple provides a framework called WKWebKit which is designed to allow easy porting of web-based information into a native application. Since Swift is such a new language, there are still significant changes that are introduced as each new version comes out. Sure enough, an update was released on April 21 that disable the previously working video stream connection, and we were left with two days to fix it before presenting.

Saving video:

This was a combination of time and changes in hardware, but we were unable to implement the feature allowing the recording and saving of videos.

Image Stabilization:

The original plan was to create a video stabilization algorithm to compensate for the rotation and wobble of a moving football. This proved to be too big of a problem to solve in the required timeframe. The algorithm would find and save a certain location of pixels and then use them to counter-rotate the image to produce a smooth result. For this to be implemented, the first step is to be able to save and extract the video footage frame-by-frame in order to be manipulated. Since this was not implemented it made the stabilization algorithm impossible to implement as well. Stabilization remains a key feature that we plan to work on in the future.

# Chapter 7: FOTOBALL PROJECT FINAL SUMMARY

Phase 1 of the Fotoball project has concluded. As expected, some key features were not implemented but the basic functionality is in place. The original plan to use the libGDX cross-platform framework to develop for Android and iOS simultaneously proved unsuccessful. We adjusted by creating an iOS app to start and are developing the Android app separately. Also, we were unable to implement the video saving aspect due to another framework and time constraints. From a hardware standpoint, we decided to start with a single camera rather than two, this time due to cost and time constraints. Finally, the issue of stabilizing a rotating image was too great a problem to solve in the required time period.

As for what was implemented, there is a prototype Fotoball, complete with an onboard camera, circuit board, wifi controller, and charging mechanism. There is also a functional iOS application capable of receiving the video being transmitted from the ball. The app is capable of running on any iOS platform of 7 or later and any iPhone from 5 or later.

In hindsight, we would likely have started with a development platform that we were already familiar with. We spent roughly three weeks learning the libGDX environment before realizing it was not suitable for our project. This was time that could have been better spent elsewhere.

It’s still unclear to us if it’s better to start programming earlier next time. The benefit of waiting until after all the design documents have been completed is you know exactly how to structure everything. Starting earlier before those documents are done provides a lot more time but will ultimately result in more changes to the code. I think it depends on your process to determine which way is better.

Overall, it was a successful endeavor into a project that really none of us had any familiarity we with. Our team meshed very well together in that we had unique skill sets for different areas of the project. This is a project that we will likely continue on with and see how far we can take it. The original discussion was for a Fotoball to be used in an NFL or major college game and we see no reasons to abandon that vision yet.

1. prepared by AbrAhAm [↑](#footnote-ref-1)
2. prepared by Rick [↑](#footnote-ref-2)
3. prepared by Adam [↑](#footnote-ref-3)
4. prepared by David [↑](#footnote-ref-4)