|  |
| --- |
| BSHCSD4 |
| Requirements Specification (RS) |
| SYNDICTR – Lotto Syndicate Mobile Application |

|  |
| --- |
| Kevin Reid – x15513377  18/11/2018 |

Requirements Specification (RS)

Document Control

Revision History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Date** | **Version** | **Scope of Activity** | **Prepared** | **Reviewed** | **Approved** |
| 18/11/2018 | 1 | Create | AB |  |  |

Distribution List

|  |  |  |
| --- | --- | --- |
| **Name** | **Title** | **Version** |
| Eamon Nolan | Lecturer |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Related Documents

|  |  |
| --- | --- |
| **Title** | **Comments** |
| Title of Use Case Model |  |
| Title of Use Case Description |  |

**Table of Contents**

[1 Introduction 3](#_Toc530344419)

[1.1 Purpose 3](#_Toc530344420)

[1.2 Project Scope 3](#_Toc530344421)

[1.3 Definitions, Acronyms, and Abbreviations 3](#_Toc530344422)

[2 User Requirements Definition 4](#_Toc530344423)

[3 Requirements Specification 4](#_Toc530344424)

[3.1 Functional requirements 4](#_Toc530344425)

[3.1.1 Use Case Diagram 5](#_Toc530344426)

[3.1.2 Requirement 1: User Login 6](#_Toc530344427)

[3.1.3 Requirement 2: User Registration 7](#_Toc530344428)

[3.1.4 Requirement 3: Create A Syndicate Group 9](#_Toc530344429)

[3.1.5 Requirement 4: Join A Syndicate Group 11](#_Toc530344430)

[3.1.6 Requirement 5: Notify Players 12](#_Toc530344431)

[3.1.7 Requirement 6: Setup Wallet 14](#_Toc530344432)

[3.2 Non-Functional Requirements 15](#_Toc530344433)

[3.2.1 Security requirement 15](#_Toc530344434)

[3.2.2 Maintainability requirement 16](#_Toc530344435)

[3.2.3 Reusability requirement 16](#_Toc530344436)

[3.2.4 Resource utilization requirement 16](#_Toc530344437)

[4 Interface requirements 16](#_Toc530344438)

[4.1 GUI 16](#_Toc530344439)

[5 System Architecture 17](#_Toc530344440)

[6 System Evolution 18](#_Toc530344441)

# Introduction

## Purpose

The purpose of this document is to set out the requirements for the development of the mobile application, **Syndicatr.** This application will allow lotto syndicates to manage their syndicate group through the medium of a smartphone.

The intended customers for this application will be mainly centred on lotto syndicate players. The main demographic that this application will be focused on is people working in companies and businesses to give them the opportunity to run and manage a lotto syndicate.

## Project Scope

The scope of the project is to develop a mobile application which will allow users to monitor and maintain their lotto syndicate groups. This application has several features which display within the application.

The aim of the application is stop users from forgetting to pay in to the lotto syndicate. Users will be able to check the status of their lotto syndicates. The application will display the lotto results for the draws that the lotto syndicates have entered. The syndicate manager will need to enter the lotto numbers into the application so that the syndicate groups will be able to see the lotto numbers that have been entered into a lotto draw. The application will automatically collect money from each player’s wallet, so they can be entered into a lotto draw seamlessly. The lotto syndicate will be alerted by app notifications if they have won an amount in a lotto draw.

To stay on schedule, a work breakdown structure and a Gantt chart was developed. By using this technique, gives the ability to track when important elements of this project need to be started and finished by.

This project will require the use of a variety of application software. The mobile application will be developed using Android Studio. Graphic design and animations will be developed in using Adobe Photoshop.

## Definitions, Acronyms, and Abbreviations

Adobe Photoshop – Image editing and graphical design software.

Android OS – Google’s mobile operating system software.

Android Studio – An integrated development software used to design and develop applications for Android OS.

# User Requirements Definition

The objective of this application to aid lottery syndicates to be managed much better. The Irish National Lottery currently does not have to option for syndicates in their own website or mobile application.

Syndicate Tool: very detailed web only application, only manages UK lottery and Euromillions draws. (https://www.syndicatetool.com/)

ELotto: subscription based lotto web application, lack of mobile application of the website, manages Irish lottery and Euromillions draws. (http://www.elotto.eu/home/default.asp)

The goal of this project is to provide lotto syndicate players the option to manage their lotto syndicate group on a mobile phone application.

# Requirements Specification

In order­ to prioritize the requirements for this project, a MoSCoW analysis chart was developed to outline which requirements will be completed and not completed during the duration of this project.



## Functional requirements

This section will outline the requirements for this project. The requirements stated will be ranked by their level of priority. The following is a list of requirements that have been ranked from the Highest to the Lowest of priority.

1. Setup Wallet: This requirement must be treated as the highest ranked requirement. The app is centered around this requirement. Much of the project time will be spent working this requirement.
2. Join A Syndicate Group: This requirement is the second highest ranked requirement.
3. User Login: This requirement is the third highest ranked requirement.
4. Create A Syndicate Group: This requirement is the third lowest ranked requirement.
5. User Registration: This requirement is the second lowest ranked requirement.
6. Notify Players: This requirement is the lowest ranked requirement.

### Use Case Diagram

This Use Case Diagram provides an overview of all functional requirements that appear in this project.



### Requirement 1: User Login

#### Description & Priority

This requirement outlines the process in which users will login to the application. To use the application, users are required to login to be able to fully use the application. This requirement is the third highest ranked requirement

#### Use Case

UC-1UserLogin

**Scope**

The scope of this use case is to allow the user to login to their user account for the “Syndicatr” application.

**Description**

This use case describes the steps in which a user will take to be able to login to their account.

**Flow Description**

**Precondition**

The system is in initialization mode.

**Activation**

This use case starts when a user must login to the application.

**Main flow**

1. The system displays the user login screen. This displays the username/email and password fields, the option for resetting a username/email or password (See A1) and the option for creating a new user account (See UC-2UserRegistration).
2. The user must enter their username/email into the input box labelled “Email”.
3. The user must enter the password into the input box labelled “Password”.
4. If the user enters in an incorrect username/email into the input box labelled “Email” (See A2).
5. If the user enters in an incorrect password into the input box labelled “Password” (See A3).
6. Once the user has both details for username/password correct, the user will be login to the application.

**Alternate flow**

A1: Resetting a password

1. If the user cannot remember the password to their account, the user will be asked to confirm their email to reset their password.
2. An email will be sent to the user which will link them to a page to reset their password.
3. The user will be asked to enter a new password which they must enter in twice to confirm that both passwords are correct.
4. The password must have at least 8 characters, consisting of lower and upper case letters and at least 2 numbers.
5. The use case continues at position 6 of the main flow.

A2: Email Validation

1. The user entered email will need to conform to a standard email address (e.g. [example@email.com)](mailto:example@email.com))
2. The email will also need to be registered within the application previously.
3. If the user cannot remember their email that they registered with the application, they will need to refer to the use case (See UC-2UserRegistration).
4. The use case continues at position 5 of the main flow.

A3: Password Validation

1. The user must put in a password that is already registered within the application previously.
2. If the user cannot remember their password that they registered with the application, they will need to refer to alternate flow A1. (See A1)
3. The use case continues at position 6 of the main flow.

**Exceptional flow**

-

**Termination**

The use case terminates when the user is logged into the application and is met with the application’s home screen.

**Post condition**

The system goes into a wait state

### Requirement 2: User Registration

#### Description & Priority

This requirement is a very important aspect of the application. User accounts are required to use this application. Users will need to be authorized prior to be granted entry to use this application. This requirement is the second lowest ranked requirement.

#### Use Case

UC-2UserRegistration

**Scope**

The scope of this use case is to allow the user to register and create an account for the “Syndicatr” application. Setting up an account will allow the user to take advantage of the application.

**Description**

This use case describes the steps in which a user will take to be allowed to use to the “Syndicatr” application.

**Flow Description**

**Precondition**

The system is in initialization mode.

**Activation**

This use case starts when the user opens the application for the first time.

**Main flow**

1. The system displays the “Syndicatr” logo to the user.
2. The system welcomes and thanks the user for choosing the application.
3. The system then asks the user if they would like to create an account for application.
4. The user either selects “Yes” or “No”. (See A1)
5. The system brings the user to the registration screen.
6. The user enters in their first name, last name, date of birth, email, a username and a password. If incorrect registration details are entered in by the user, the screen reloads. (See A2)
7. The system greets the user to say their registration is complete and asks the user to check their email to confirm their email address.
8. The user confirms their email address.
9. The user registration process is finished.

**Alternate flow**

A1: User chooses “No”

1. The system tells the user that registration is required to use the application.
2. The user is asked if they want to “Create an account” or “Leave the application” (See E1)
3. The user chooses the “Create an account” option
4. The use case continues at position 5 of the main flow.

A2: Registration Validation

1. If the user enters details the application doesn’t accept, the user will need to amend the required details.
2. The user entered email will need to conform to a standard email address (e.g. [example@email.com)](mailto:example@email.com))
3. The user name received by the user must not be already been entered by another user.
4. The password must have at least 8 characters, consisting of lower and upper case letters and at least 2 numbers.
5. The age requirement of this application is 18 years of age. Any users under this age will not be given access.
6. If the user fulfils all these requirements
7. The use case continues at position 7 of the main flow.

**Exceptional flow**

E1: User chooses “Leave the application”

1. The user has chosen to “Leave the application”.
2. The system closes the application.
3. The user is return to the mobile device’s home screen.

**Termination**

The use case terminates after the user confirms their email address and returns to the application.

**Post condition**

The system goes into a wait state

### Requirement 3: Create A Syndicate Group

#### Description & Priority

This requirement outline the process in which a syndicate group is set up within the application. This requirement is the third lowest ranked requirement.

#### Use Case

UC-3CreateASyndicateGroup

**Scope**

The scope of this use case is to allow the user to create a syndicate group within the “Syndicatr” application.

**Description**

This use case describes the process in which a user will take to set up a new syndicate group within the application “Syndicatr”. The user will need to consider a few details to setup a new syndicate group.

**Flow Description**

**Precondition**

The system is in initialization mode.

**Activation**

This use case starts when a user chooses the “Groups” option in the in-app menu.

**Main flow**

1. The user has chosen the “Groups” option in the in-app menu.
2. The user will be given the option either create a syndicate or join a syndicate. (See UC-4JoinASyndicateGroup)
3. The user chooses the option, “Create a syndicate”.
4. The system will bring the user to the “Create a new syndicate” page.
5. The system will ask the user to enter the details of the syndicate and it will look for the user to enter in the following fields: syndicate name, emails of syndicate users, date in which to remind users to pay in and a description field to display the syndicate rules.
6. The system will auto assign the user who has created the syndicate group to role of syndicate manager.
7. If the user would like to assign another user the role of syndicate manager, then they must wait until that user has been enrolled into the group.
8. Once these details have been addressed, the user must click the button labelled “Create Group”.
9. The system will be sent the user back to the “Groups” screen and the new syndicate group will have been generated.

**Alternate flow**

-

**Exceptional flow**

-

**Termination**

The use case terminates when the user has created the syndicate group and the user is brought back to the “Groups” screen.

**Post condition**

The system goes into a wait state.

### Requirement 4: Join A Syndicate Group

#### Description & Priority

This requirement outlines the process in which a user join an existing syndicate group already created by another user. This requirement is the second highest ranked requirement.

#### Use Case

UC-4JoinASyndicateGroup

**Scope**

The scope of this use case is to allow the user to join a pre-existing syndicate group which was set up by another user.

**Description**

This use case describes the steps in which a user will take to join a syndicate group within the “Syndicatr” application. The user will need to be approved by the Syndicate manager to be able to join the group.

**Flow Description**

**Precondition**

The system is in initialization mode.

**Activation**

This use case starts when a user chooses the “Groups” option in the in-app menu.

**Main flow**

1. The user has chosen the “Groups” option in the in-app menu.
2. The user will be given the option either create a syndicate (See UC-3CreateASyndicateGroup) or join a syndicate.
3. The user chooses the option, “Join a syndicate”.
4. The system will bring the user to the “Join a syndicate” page.
5. The system will ask the user to enter a unique syndicate password.
6. The user should have received this password from the Syndicate Manager to group that they wish to join.
7. The user will be prompted if the password they entered is incorrect. (See A1)
8. Once the user has entered the syndicate password in correctly, the system will allow them access to the syndicate group.
9. The system will be sent the user back to the “Groups” screen and the syndicate group that wish to join has been generated.

**Alternate flow**

A1: Incorrect Syndicate Password

1. If the user has entered a password which the system detects is incorrect, the user must try and re-enter the password and make sure they put it in correctly.
2. If the password is incorrect again, the user must contact the Syndicate Manager of the group to get them to re-send the password again.
3. The use case continues at position 8 of the main flow

**Exceptional flow**

-

**Termination**

The use case terminates when the user has joined the syndicate group and the user is brought back to the “Groups” screen.

**Post condition**

The system goes into a wait state.

### Requirement 5: Notify Players

#### Description & Priority

This requirement describes the process of notifying syndicate groups, syndicate managers and syndicate players. This requirement is the lowest ranked requirement.

#### Use Case

UC-5NotifyPlayers

**Scope**

The scope of this use case is to notify syndicate players to carry out various tasks and to notify of any changes to syndicate groups.

**Description**

This use case describes the steps in which the system will notify players, managers and groups on payment notifications, lotto payouts and general updates.

**Flow Description**

**Precondition**

The system is in initialization mode.

**Activation**

This use case starts when the system sends a notification to the user’s mobile phone.

**Main flow**

1. The user will receive a notification from the application from various situations.
2. If the user needs to top-up their in-app wallet, the user will be prompted by a notification. (See A1)
3. If the user notified to acknowledge a lotto pay-out, the user will be prompted by a notification. (See A2)
4. If the user notified by new syndicate group update, the user will be prompted by a notification. (See A3)
5. Once the user has acknowledged any of these notifications they can proceed as normal.

**Alternate flow**

A1: Notification to top-up wallet

1. If the user is prompted to top-up their wallet, they will be prompted because of two reasons: if they have currently have no money in their in-app wallet or the syndicate groups in which they are entered into has request them to pay in to be entered into the lotto draw.
2. The user will then need to open the application.
3. The user will need to proceed to the wallet screen from the main menu.
4. The user will need to choose the option for “Top-Up Wallet”.
5. As the user will already have setup their wallet (if they have not see: UC-6SetupWallet), they will need to choose the amount they wish to top-up by and last 3 digits on the back of their payment card.
6. The user then must choose the option to “Top-Up now”.
7. The system displays to the user that their wallet has now been topped up.
8. The use case continues at position 5 of the main flow.

A2: Acknowledge lotto pay-out

1. If the user is notified that a lotto payout has occurred, the user must open the application to find out more details.
2. Each user of syndicate group will be notified if a lotto payout has occurred.
3. The syndicate manager will organize the payout of the potential winnings with the syndicate group outside of the application.
4. The use case continues at position 5 of the main flow.

A3: Syndicate updates

1. If the user is notified that an update has occurred to one of the syndicate groups that they have entered in to, they will be asked to acknowledge the update.
2. The syndicate update will usually pertain to a message from the syndicate manager, a user has join or left the syndicate group or syndicate rule has been changed or amended.
3. The use case continues at position 5 of the main flow.

**Exceptional flow**

-

**Termination**

The use case terminates when the user has acknowledged the notifications that the application has sent to the user.

**Post condition**

The system goes into a wait state.

### Requirement 6: Setup Wallet

#### Description & Priority

This requirement describes the process in which the user sets up the in-app wallet that is needed to allow the user to enter in lotto draws. This requirement must be treated as the highest ranked requirement. The app is centered around this requirement. Much of the project time will be spent working this requirement.

#### Use Case

UC-6SetupWallet

**Scope**

The scope of this use case is to describe the process in which user sets up the in-app wallet for the “Syndicatr” application.

**Description**

This use case describes the steps a user will take when they will need to setup the in-app wallet to enter lotto draws within the syndicate groups.

**Flow Description**

**Precondition**

The system is in initialization mode.

**Activation**

This use case starts when the user chooses the option for “Payments” in the main menu.

**Main flow**

1. The user must go to the option for “Payments” in the main menu.
2. In the “Payments” menu, the user must enter the following details: Cardholder name, Card number, Expiry date and Security code.
3. The user can only enter one payment option per account. If they wish to add another card to the account, they must remove delete the previous card details.
4. When the user has entered these details, they must then proceed to add the card by choosing the “Add” button.
5. The user’s card will now be assigned the application’s payment options.
6. The system will acknowledge that the card has been successfully entered.

**Alternate flow**

-

**Exceptional flow**

-

**Termination**

The use case terminates when the system tells the user that the card has been successfully entered.

**Post condition**

The system goes into a wait state.

## Non-Functional Requirements

### Security requirement

User’s personal details will need to be stored securely. To be able to make sure user data remain safe, the application will encrypt this data using robust encryption software.

### Maintainability requirement

Updates should be as infrequent as possible. Scheduled maintenance should be announced days prior. Maintenance will be scheduled to times of the day in which the app is least used.

### Reusability requirement

The application enables users to easily maintain the role within a syndicate groups. While screen time within the application should be relatively low, users will find the application useful when they receive notifications from the application itself.

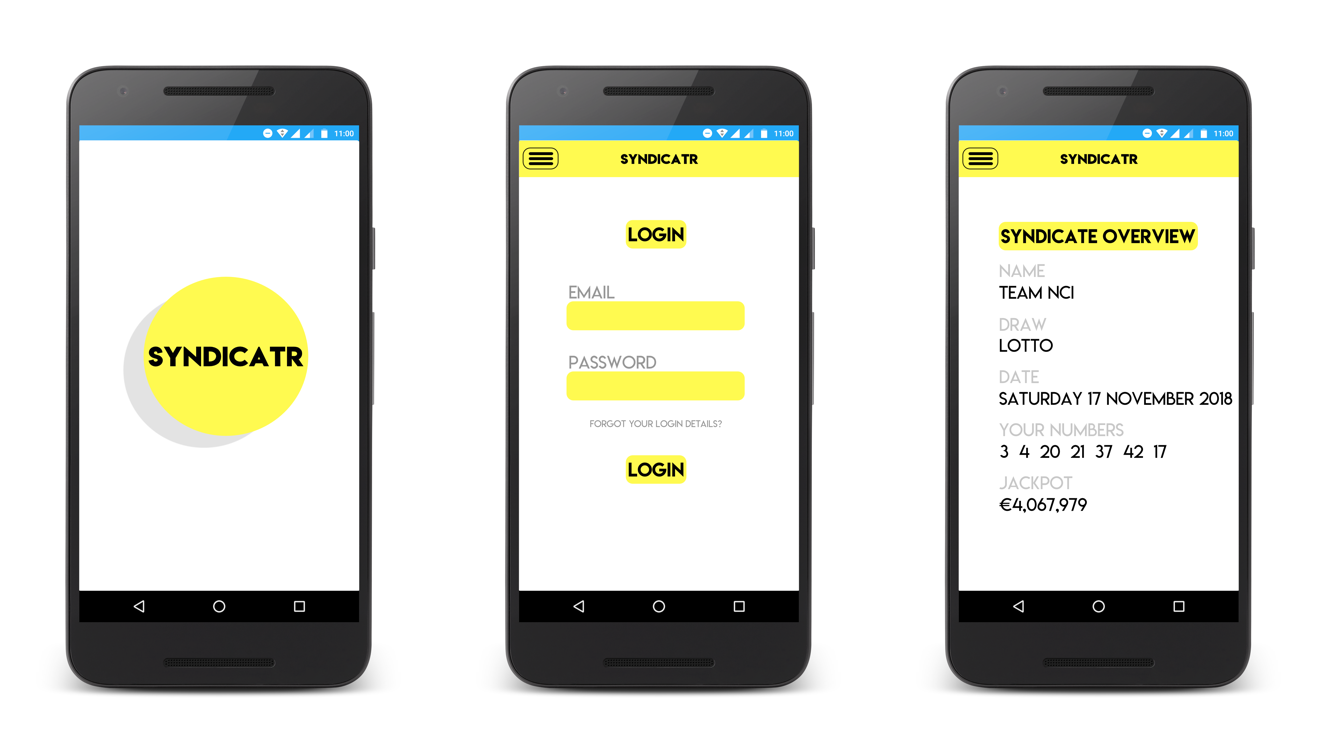
### Resource utilization requirement

This application does have software and hardware requirements. If users wish to download this application their mobile phone must be at least running Marshmallow 6.0 for Android devices. A constant internet connection is required for using this application. Ideally a 4G connected phone would be the preferred choice of device for using this application.

# Interface requirements

## GUI

Each of the GUI mock-ups displayed show how the application will provisionally look at this stage of the project. These designs have been commissioned to showcase the GUI and color scheme of the application.





# System Architecture

A class diagram was developed to show the relationships between the different architectures that are present in this application. The four areas which have been outlined in the class diagram below are the wallet, the players, managers, groups and draws.

Below is a list of the relationships that found in the class diagram.

* Syndicate Groups can one or more players
* Players can have one or more Syndicate Groups
* Players can only have one Wallet
* Syndicate Groups can one or many lotto draws entered.
* Syndicate Groups can have only one Syndicate Manager



# System Evolution