**COMP 3059 – Capstone Project I**

**Software Requirements Analysis and Design Assignment**

This assignment is an overview to gather the software needs with requirements analysis and help to proceed with the design.

The requirements analysis helps to break down functional and non-functional requirements to a basic design view to provide a clear system development process framework. It involves various entities, including business, stakeholders and technology requirements.

The design is the activity following requirements specification and before programming. Software design usually involves problem solving and planning a software solution.

To work on this assignment you could use the references and a sample template given below. The sample template can be customised to suit the nature of your project.

Reference Readings/Example:

<http://www.uacg.bg/filebank/acadstaff/userfiles/publ_bg_397_SDP_activities_and_steps.pdf>

[www.cse.msu.edu/~chengb/RE-491/Papers/SRSExample-webapp.doc](http://www.cse.msu.edu/~chengb/RE-491/Papers/SRSExample-webapp.doc)

<https://nces.ed.gov/pubs2005/tech_suite/part_2.asp>

Reference template:

[www.tricity.wsu.edu/~mckinnon/cpts322/cpts322-srs-v1.doc](http://www.tricity.wsu.edu/~mckinnon/cpts322/cpts322-srs-v1.doc)

# 1.0 Introduction

The purpose of this document is to outline and describe the high level requirements related to the “How To Train Your Dragon Boat” Application. The sections below will describe the systems features and the purpose of them, as well as describing their functionality through a use of diagrams. This requirements associated with the application are not limited to, but will include the functional and non function requirements of the system, and database requirements for the application.

## Purpose

The Main Purpose of the “How To Train Your Dragon Boat” Application is to provide the George Brown College Dragon Boat Team with an application that is intended to aid the team with the common tasks performed for the team. The team’s most important tasks, which include the management of team members, the creation of dragon boat layouts, and the monitoring of dragon boat teams during practice are currently done through manual methods, mainly through scrap paper, and as a result are of very low effectiveness and efficiency. The How To Train Your Dragon Boat application will tackle these specific problems, in order to provide a much higher level of efficiency during the completion of the Dragon Boat team’s tasks.

## Scope

* **In-Scope Specifications:** The How to Train Your Dragon Boat Application will be a mobile application supportable by both iOS and Android operating systems. It will also be supported on tablet platforms as well. The application will provide an interactable interface for the creation of customization of boat layouts, with the ability to assign users to the layout. Details about the current layout, including the weight distribution of the members, or their seating preferences, will be displayed along with the layout. This requirement replaces the current method of constructing a layout, which is performed manually, by providing a semi-automated system which will increase efficiency. A map interface will be accessible within the application, designed to assist the team with a valid method of tracking the route in which the boat pursued, as well as the performance of the boat during the current run. Details of the boat’s performance include the distance the boat travelled, the speed or velocity of the boat, and the boat layout that was used during in the current run.

The application will also provide a CRUD interface to manage the users of the dragon boat team, or any other user that may or may not be associated with the dragon boat team. Communication can also be established between users through the use of associated email addresses, eliminating the need to communicate through third party sources or face-to-face conversation. Provided within the application will be an SQLite database that will store all of the users, routes, boat layouts, and the associated information related to each item stored. This will provide a central storage that can be easily accessed by the application and its users.

* **Out-Of-Scope Requirements:** Support for the application will not extend beyond the mobile scope, and will not be supportable by any operating system beyond Android or iOS systems. While the application will use GPS and location for the functionality of the map interface, the application will not use any internet connectivity for functions of the application. The map provided with the application will be used for location tracking only, and will not include any form of custom paths for boat routes. In addition, the map will not include any form of location finding. To be specific, this application will not allow users to use the GPS to find a specific location within an area.

\*(Highlighted statements are new additions to the In Scope/Out or Scope Requirements)

# System Overview

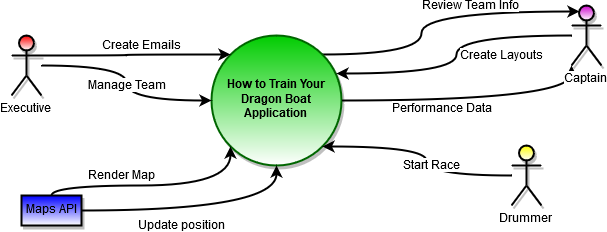
## Project Perspective

Currently, the George Brown College Dragon Boat team does not have any current system software for the management of their tasks, forcing the team to perform these tasks manually. As a result, these tasks prove to be both unorganized and time consuming. The How To Train Your Dragon Boat application will be built upon resolving these issues, and will serve a new self-contained system for the team.

## System Context

The How To Train Your Dragon Boat application has four actors interacting with the system, which include 3 users and one software API. The Executive is responsible for managing any members of the team, including adding new members, altering current members, or deleting old members. The members will be stored in an internal database table contained within the application, The executive can also communicate with any of the team members by email. The captain is responsible for using the list of team members to create boat layouts that will be used during practice runs. The layouts will be created using an interactive drag and drop interface and will be stored within the application’s internal database similar to the team members. The drummer’s role is to use the boat layouts created by the captain to begin tracking the boat statistics during practice runs. More specifically, the drummer will responsible for beginning the application’s tracking function. The application will record the boat position and route through a map API, and also the distance the boat travelled, and the velocity of the boat. This information will be stored as routes within the database to be viewed by the team captain. Finally, the map API is part of the tracking process that is initiated by the drummer. It is responsible for the updating and positioning of the boat as it travels through the map. This information will be stored along with the performance information inside the database.

## System Context Diagram



## General Constraints

Specification Constraints:

* Current goal of production and development involves little to no budget costs or expenses. Results in a much smaller selection of available software IDEs for development.
* Deadlines set for project components / sprints are not flexible, and must be accomplished by the set due date.
* Tasks defined for the project and the division of the tasks is limited by low team size

Designing Constraints:

* Wireframes created must keep any limitations of mobile development in mind when designing components, appearance, or basic navigation

Implementation constraints:

* Implementation of database schema have the potential to vary based on the IDE selected for development
* The use of, or the implementation of the React Native Npm library will vary depending on the selected IDE for development
* Appearance or functionality of application may hold slight differences to the wireframes or diagrams depending on the software IDE’s limitations or the development work required.

Testing constraints:

* Testing GPS in real life conditions relies on access to boat and weather conditions
* Database storage testing, while unlikely, is dependant on the available storage the testing device currently has.
* The target device for testing must meet the software version requirements for the application to run
* The target device must have all required permissions enabled for all functionality to be tested

## Assumptions and Dependencies

List any assumptions that have been made during the initiation of the project. In addition, list any dependencies that may impact its success or the desired result.

Assumptions:

* Projects members are willing to assume different roles to make up for the lack of members involved in the project and to assist members in the completion of tasks
* A consistent form of communicate will be obtained across all team members despite each member’s involvement in educational activities
* Project members agree to follow the guidelines, requirements, and plans that are discussed and written in the documentation
* Project members are aware of the risks of not meeting specified deadlines.
* Project members are willing to research new languages and technologies for the project if needed
* Any sections labelled in the document including scope requirements, constraints, dependencies, features, diagrams are not final and may concur changes as further development of the project continues.
* The online marketplace will be able to support the application as it is distributed to its users
* The GBC Dragon Boat Team will have limited involved in the development of the application.
* The React Native libraries and framework being used for this project will continue to be supported by Facebook
* The chosen map API that the application will use continues to provide free support
* The database chosen for the application will be continuously provided for mobile development
* The Users of the application will meet the requirements to run the application on their mobile device.

Dependencies:

* Non-Functional Dependencies:
* Users of the application must have location enabled for the map API to function properly
* Access to an iOS device or software is needed to test iOS support
* Access to an Android device or software is needed to test Android support
* Functional Dependencies \*\*:
* Npm install packages
* React native npm library
* React native navigation npm library
* React native drag-and-drop npm library
* React native maps npm library
* React native animation npm library

\*\*For simplicity, only major library dependencies have been listed, as actual number of dependencies needed is of a much larger quantity

## 3.0 Functional Requirements

|  |  |
| --- | --- |
| **Use Case Name** | Add Boat Layout |
| **Trigger** | The User selects to add a new boat layout to the database |
| **Precondition** | The User has accessed the Layouts screen |
| **Basic Path** | 1. The user clicks the “Add” button displayed on the layouts screen 2. The system presents a new screen where the user is prompted to give a layout name and layout size 3. The user decides if they want to use a predetermined size layout, a custom layout or a previously used layout 4. Upon selection the layout has been created and entered into the database |
| **Postcondition** | The database has been updated. |
| **Exception Paths** | The User may abandon the operation anytime |
| **Other** | The layout information includes the name of the layout and the number of seats to be included |

|  |  |
| --- | --- |
| **Use Case Name** | View Boat Layout |
| **Trigger** | The User selects a Layout on the Layouts screen |
| **Precondition** | The User has accessed the Layouts screen |
| **Basic Path** | 1. The user selects the desired layout on the Layout screen |

|  |  |
| --- | --- |
| **Use Case Name** | Edit Boat Layout |
| **Trigger** | The User selects to edit a selected Boat Layout |
| **Precondition** | The User has accessed the Layouts screen and has selected a Layout to view. |
| **Basic Path** | 1. The user selects the boat layout they wish to edit 2. The user can drag paddlers existing paddlers on the layout onto different positions 3. The user can open a sliding drawer menu with a repository of existing paddlers and can filter through by name and position 4. The user can drag selected paddlers onto the layout 5. The user saves the layout and the information is stored in the database |
| **Postcondition** | The database and layout have been updated. |
| **Alternative Paths** | In Step 2 the user can have a sorting algorithm automatically assign paddlers onto the boat depending on weight and optimal weight distribution onto the boat. This is only if Paddlers exist onto the boat  The user can also edit the name of a layout |
| **Exception Paths** | The User may abandon the operation at any time |

|  |  |
| --- | --- |
| **Use Case Name** | Delete Boat Layout |
| **Trigger** | The User selects to delete a selected Boat Layout |
| **Precondition** | The User has accessed the Layouts screen and has selected a Layout to view. |
| **Basic Path** | 1. The user selects the Layout they wish to delete 2. The user selects the delete option and is given a confirmation prompt 3. If selected yes, the selected layout is deleted |
| **Postcondition** | The layout has been deleted and the database has been updated |
| **Alternative Paths** | The user can also delete a layout from the Layouts screen. By holding a layout to enable the delete prompt in the Layouts screen and selecting a layout to be deleted |
| **Exception Paths** | The User may abandon the operation at any time |

|  |  |
| --- | --- |
| **Use Case Name** | Search for Boat Layout |
| **Trigger** | The User chooses to search for a Boat Layout in a search bar |
| **Precondition** | The User has accessed the Layouts screen |
| **Basic Path** | 1. The user selects the search navigation bar 2. The user can filter layouts by size 3. The user enters the name of the layout they wish to find |
| **Postcondition** | The user is given a filtered list of all layouts that match their criteria |
| **Exception Paths** | The User may abandon the operation at any time. If there are no existing layouts, the search option will return nothing. |

|  |  |
| --- | --- |
| **Use Case Name** | Add Teammate |
| **Trigger** | The User selects to add a new teammate to the database |
| **Precondition** | The User has accessed the Team screen |
| **Basic Path** | 1. The user selects the “Add” button displayed on the Team screen 2. The system presents a screen with empty text fields regarding teammate information 3. The user enters the appropriate information into the fields 4. The system checks if the name field is not blank and adds them to the database |
| **Postcondition** | A teammate has been added and the database has been updated |
| **Exception Paths** | The User may abandon the operation at any time |
| **Other** | Teammate information includes, name, email address, phone number, weight, height, age, if they’re a paddler, and paddling side preference, |

|  |  |
| --- | --- |
| **Use Case Name** | Edit Teammate |
| **Trigger** | The User selects to edit an existing teammates information |
| **Precondition** | The User has accessed the Team screen |
| **Basic Path** | 1. The user selects a teammate they wish to edit in the Team screen 2. The user views the teammates information in their details screen and selects the “Edit” option 3. The text fields become editable and the user edits the information they wish to modify 4. The user clicks the save option |
| **Postcondition** | The teammates information has been modified and the database has been updated. |
| **Exception Paths** | The User may abandon the operation at any time |
| **Other** | This use case can be used to add categories for an article, to correct typographical errors, or to remove a reviewer who has missed a deadline for returning a review. It may also be used to allow access to the named use case to enter an updated article or a review for an article. |

|  |  |
| --- | --- |
| **Use Case Name** | Delete Teammate |
| **Trigger** | The User selects to delete a teammate to the database |
| **Precondition** | The User has accessed the Team screen |
| **Basic Path** | 1. The user selects a teammate they wish to delete from the database 2. The user selects the edit option to change the accessible fields 3. The user selects the newly shown delete button 4. The user is given a prompt to confirm their choice and selects yes to confirm the delete operation |
| **Alternative Paths** | The user can also delete a teammate from the Team screen. By holding a teammate to enable the delete prompt in the Team screen and selecting the teammate to be deleted |
| **Postcondition** | The user has been deleted and the database has been updated. |
| **Exception Paths** | the User may abandon the operation at any time. |

|  |  |
| --- | --- |
| **Use Case Name** | Search for Teammate |
| **Trigger** | The User selects to search for a teammate in the Team screen |
| **Precondition** | The User has accessed the Team screen |
| **Basic Path** | 1. The user selects the search navigation bar 2. The user can filter teammates by name, paddling side preference, weight and height 3. The user enters the name of the teammate they wish to find |
| **Postcondition** | The user is given a filtered list of all teammates that match their criteria |
| **Exception Paths** | The User may abandon the operation at any time. Additionally, if there are no teammates in the database the search will retrieve nothing |

|  |  |
| --- | --- |
| **Use Case Name** | View Performance Analytics Chart |
| **Trigger** | The User selects to a performance chart to view |
| **Precondition** | The User has accessed the Analytics screen |
| **Basic Path** | 1. The user selects the desired chart in the Analytics screen |
| **Postcondition** | The user is routed to a screen displaying performance analytics about boat layouts for their selected chart |

|  |  |
| --- | --- |
| **Use Case Name** | Search for Performance Analytics Charts |
| **Trigger** | The User selects to add a new boat layout to the database |
| **Precondition** | The User has accessed the Team screen |
| **Basic Path** | 1. The user selects the search navigation bar on the Team screen 2. The user can filter charts by layouts, name, date and rating 3. The user can search by name of the chart |
| **Alternative Paths** | In Step 1 the user can be redirected from viewing a Boat Layout to searching for a performance chart. The search filter will be set to the information regarding the selected layout and information regarding that layout will be shown |
| **Postcondition** | A displayed filtered list of analytics charts depending on the search criteria |
| **Exception Paths** | The User may abandon the operation at any time. Additionally, if there are no charts in the database the search will retrieve nothing |

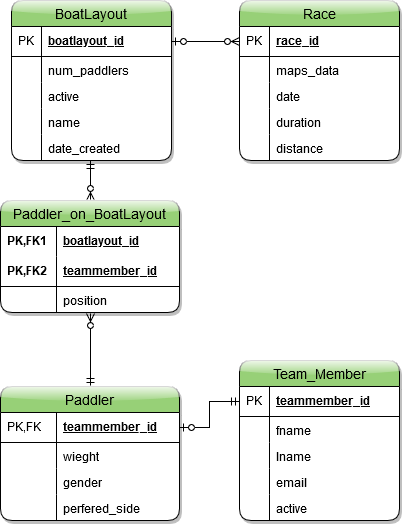
|  |  |
| --- | --- |
| **Use Case Name** | Delete Performance Analytic Chart |
| **Trigger** | The User selects to chart to delete from the database |
| **Precondition** | The User has accessed the Analytics screen |
| **Basic Path** | 1. The user selects the chart they wish to delete 2. The select the options menu in the top right and select the “delete” option 3. The user is given a prompt confirming if they wish to delete the chart |
| **Alternative Paths** | The user can also delete a chart from the Analytics screen. By holding a chart to enable the delete prompt in the Analytics screen and selecting the chart to be deleted |
| **Postcondition** | The chart has been deleted and the database has been updated. |
| **Exception Paths** | The User may abandon the operation at any time. |

|  |  |
| --- | --- |
| **Use Case Name** | Create Performance Analytics Chart |
| **Trigger** | The User given a prompt to save a chart |
| **Precondition** | The user has completed a route on the Ready screen and has GPS permissions enabled |
| **Basic Path** | 1. The user is given a prompt to save a performance analytics chart for the just completed boat route 2. The user can choose to save the route or not to save it 3. If the user saves, the system displays a text field to give the chart a name |
| **Postcondition** | The user saves a performance chart, the database will be updated with that information |
| **Exception Paths** | The user may abandon the operation at any time. If the user has not enabled GPS permissions, they cannot access this use case |

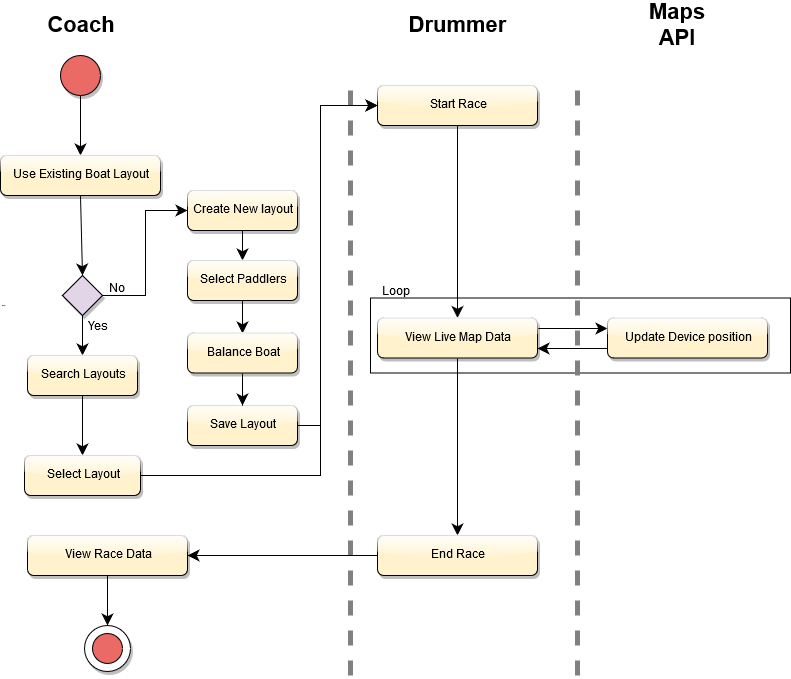
|  |  |
| --- | --- |
| **Use Case Name** | Record GPS Route |
| **Trigger** | The User selects the Go button on the Ready screen |
| **Precondition** | The User has accessed the Ready screen and has enabled GPS permissions in the application |
| **Basic Path** | 1. The user selects the layout they wish they use to record GPS data for 2. The user selects the start button to begin recording data 3. When the user decides to stop recording data they can press the stop button 4. Once the recording is stopped the user is given a chance to save a performance chart of the GPS data for that layout |
| **Alternative Paths** | In step 3 the user can press the pause button to optionally stop or restart recording data. And can further press stop again to stop for good.  In step 1, the user can choose not to use a layout and record a GPS route anonymously and save the chart as an anonymous layout |
| **Postcondition** | If the user chooses to save a performance chart, the database will be updated with that information |
| **Exception Paths** | The user may abandon the operation at any time. If the user has not enabled GPS permissions, they cannot access this use case |

**3.3 Data Modelling and Analysis**

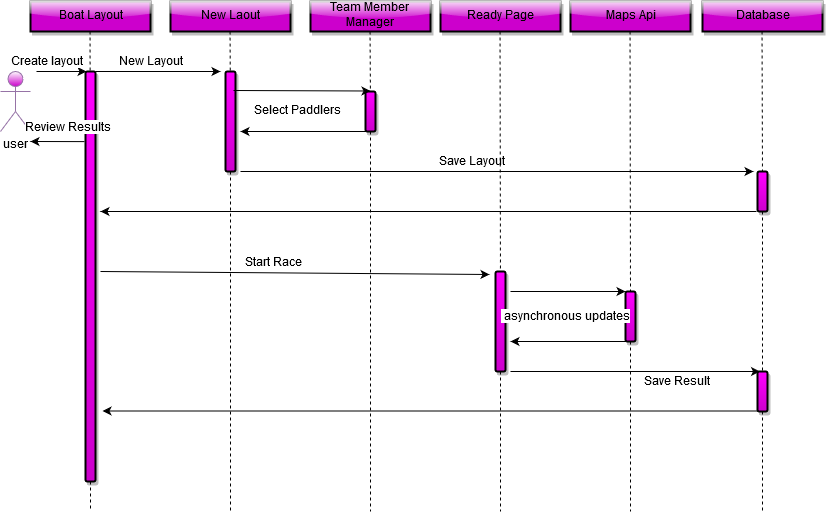
* Normalized Data Model Diagram



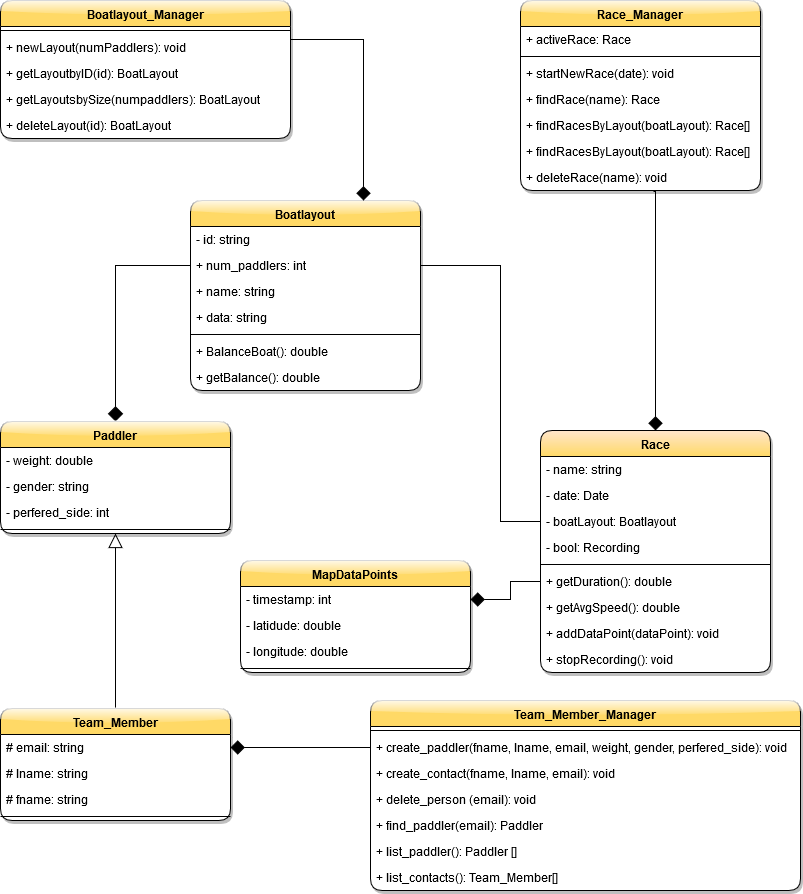
* Activity Diagram



* Sequence Diagram

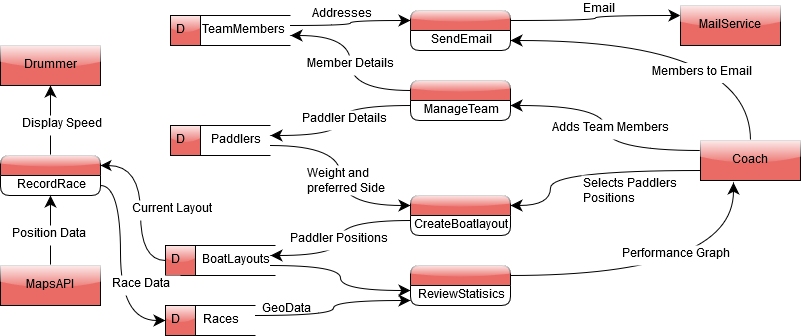


* UML Class Diagram



**3.4 Process Modelling**

Data Flow Diagram



## 4.0 Non-Functional Requirements

The How to Train your Dragon Boat application can be installed on any Android device running Android 4.1 or higher and any IOS device running IOS 9.0 or higher.

The software developed uses built in GPS features for mobile devices. The speed and accuracy of the GPA data depends on the hardware specifications of the individual device as well as the conditions that affects GPS data from the physical location of the device being used.

In app speed and performance depends on the hardware and device operating system being used on the device. More modern devices and system OS’s will perform better in regular app usage.

The application in its current scope does not require internet access and does not store any private/confidential information.

## 5.0 Logical Database Requirements

A local database will be used to store information on team members as well as past performance in races and practices. As this information is stored locally on the users phone, file size is a concern, as such map data will be stored in the database but other analytics (eg. average speed, speeds consistency) will not be store in the data base as they can be recalculated from the original data. The database will be required to be performant on both Android and iOS so that the user experience is both responsive and consistent across both platforms. As an internet or cloud storage is currently out of scope there will be no opportunity to create multiple or offsite backups for this data, while this is a disadvantage it is acceptable in this case as the main focus of the app is to manage the current race and give live performance feed back. If in a later release we decided to increase the focus on data analytics of this app we will also have to explore adding internet storage and backup to the app.

## 6.0 Other Requirements

Additional requirements, if any.

**7.0 Approval**

The signatures below indicate their approval of the contents of this document.

|  |  |  |  |
| --- | --- | --- | --- |
| Project Role | Name | Signature | Date |
| Systems Architect | Andrew Cobb |  | 11/11/2019 |
| Scrum Master | Giuseppe Ragusa |  | 11/11/2019 |
| Project Manager | Arsalan Farooqi |  | 11/11/2019 |