Wavelength Sail Manager

Luke McIlwaine – B00767397

Project Supervisor – David Glass

2022

Table of Contents

**No table of contents entries found.**

# Introduction

## Background

In sail racing, every class or model of boat will have different performance characteristics. To enable different classes to compete against each other in the same race, a handicap scaling factor is applied rather than simply first across the line. This scaling factor is a universally recognised standard as defined by the Royal Yachting Association.

These ‘true’ results must be calculated at the end of the race by hand. This exposes many inadequacies of the system as the Race Officer must first record the type of craft racing and their handicaps. The 5-minute sailing counter must then be started and the lap and finish times recorded as well as process special classifications e.g. DSQ (Disqualification). After the race, the true times must be calculated and input into a database which may then be read at different times throughout the year. This process is repeated manually multiple times each day.

My project is intended to eliminate these tasks by creating an intuitive race management program operating on a tablet or touch screen device whereby the manual effort is removed. This is likely to help a wide variety of race officers and in particular, those who are new to the role or may be less tech-literate. In addition, it will significantly reduce the possibility of error.

There will be many technical challenges with this project, particularly in relation to the storing and manipulation of very complex customisable data. A multitude of administrative features must also be implemented providing the ability to add sailors, classes and configure races amongst others.

A useful feature that should be implemented is the use of serverless technologies to generate data about the weather and current location. These are pieces of information which will be useful for administrators and race officers.

## Aim

In this project I aim to create an all-in-one handicap racing solution where the user starts and manages the race as well as calculating the racing results and providing data manipulation tools for each series of races.

## Objectives

* Create a mechanism for building a library of boat characteristics.
* Create an efficient way for the user to add boats that are racing to a timing screen.
* Allow for the 5-minute sailing countdown to be triggered, reset and managed to remove the need for a clock.
* Provide the ability to calculate results including being able to base off lap times should the race be shortened.
* Give the ability to finish boats easily and calculate the results acknowledging the handicap.
* Display and store the results for future use and administration.
* Make the user processes as simple as possible to ensure widespread usability.
* Provide flexibility for administrators to configure the application in a way that is suitable for the club.

# Project Review and Tecnhical Investigations

## Stakeholder Investigation

## Requirements and analysis

### Requirements Gathering techniques

#### Primary stakeholder

In my project I selected to have one primary stakeholder. This is primarily because I have substantial experience in this area myself. I choose Liz as my primary stakeholder due to her numerous years as chief race officer for Newcastle Yacht Club. As part of this she has invested significant amounts of time working with a similar but distinct application, HalSail. Liz also has vast experience as a race officer working with multi-class fleets of sailing dinghies.

##### Primary stakeholder strategy

My primary method for gathering resources and requirements from the primary stakeholder is an interview style format. The focus of this initial interview is to gather requirements of the ideal program. This is likely to be specific to the needs of a small club, this is ideal as it’s the programs intended purpose. I also plan to gain information around the way Hal is used within the club and gather the improvements that could be made. Unfortunately, neither Liz nor anyone else I know has used the mobile version, likely due to its very limited support. Due to that I plan to discuss the app with Liz and perform brief usability testing to gather the immediate issues.

The purpose of this interview is also two-fold, one part to gain requirements and the other to understand common issues within existing applications and aim to fix them. The requirements will form a significant proportion of the information gathering session, by the end I will hopefully have a set of comprehensive notes which I can turn into requirements.

##### Planned 1st stage questions

* Can you give a description of managing a race from the perspective of a race officer when not using software to assist and working with everything manually?
* Can you give a description of managing a race from the perspective of a race officer when using software to assist (HAL)?
* Can you tell me why you prefer one option over the other?
* How do you normally organise the timing aspect of the race?
* Have you ever used a software alternative to manage the timing aspect of the race?
* Do you have any frustrations with the timing aspect of the race currently?
* Do you have any frustrations with the entering of race data into HAL?
* You are an administrator of HAL, how friendly do you find the use of the application from administrators’ perspective?
* Are you happy with the way Hal manages the data?
* Can you think of any immediate improvements you would make with HAL?
* Here’s a mobile application with timing functionality, take a look around and play with the app.
* Would you find using an app that also records the data for a series useful?
* Do you have any improvement suggestions for the app?
* Finally, is there anything you would like to see in a future application used to manage both the timing aspect as well as the data handling.

##### Main stakeholder Interview

An hour-long interview was conducted early in the refinement gathering process. More information on the questions to be asked is included in add location here. The primary stakeholder is the chief race officer at Newcastle YC. Liz manages all aspects of racing and as such will have a clear view of the race administrator process. Through this interview key takeaways were gained which will heavily influence the requirements that are set and the prioritisation of each. An example of some of the key items gathered that previously hadn’t been discovered through analysis are shown below:

* The membership secretary should be able to maintain a list of the racing craft and sailors. This is important to allow those who have access to the information to set the correct information.
* Race officers generally prefer to be on the water. As such the software could be on a portable waterproof device to assist with this.
* Can use a stopwatch style timing system rather than a current time calculation. Allows for more flexibility.
* PY calculation should be flexible and adjust for different numbers of laps.
* Include buttons to lap and record lap times. The number of laps completed should be included to increase race officer understanding.
* Allow assigning of special values at any point in the race.
* Add series calculation functionality to show winners and results per series.

#### Secondary stakeholders

##### Secondary stakeholder strategy

Although I have a key primary stakeholder, I would still like to include further officers to ensure that I have accurately captured the end user and not just the administrator’s perspective. For this I will create a survey which will be shared within the main group of people who share race management responsibilities, the survey questions can be found below:

* When performing race officer duties do you prefer to use pen and paper to calculate the results or software on the laptop? \*
* How many times in a year are you likely to be race officer or interact with series management? \*
* Rate the current difficulty of starting, finishing and timing a race?
* What are the top 3 improvements you would make to the race management from a race officers perspective?
* Do you normally interact with a data recording system when on duty (e.g. HalSail, SailWave) \*
* If yes or sometimes, rate how easy that application is to use and understand.
* Have you ever used an app to manage the timing aspect of the race? e.g. Boat Timer
* How likely are you to use the following features if they are available?
  + Add boats to that day’s racing by being able to select them from a list rather than asking each individual? \*
  + Start a race at any time by pressing a button which provides a countdown rather than starting a race based on a specific time? \*
  + Record the finish time of each boat when they cross the line by pressing a button, where the handicap time is then calculated automatically? \*
* Are there any other features you may find useful?

##### End user questionnaires

End user questionnaires were sent out to a variety of individuals who have interactions with race management on a regular to semi-regular basis. In total 6 individuals responded, this is a good number for small club. The questions asked can be viewed at insert section.

The results received were useful to corroborate the views of the chief race officer however two questions asked provided more interesting results. These were ‘Have you ever used an app to manage the timing aspect of the race? E.g. Boat Timer’ (Q1) and ‘When performing race officers duties do you prefer to use pen and paper to calculate the results or software on the laptop’ (Q2).

The results received from Q1 (Fig 1) show a clear gap in the market to introduce a mechanism for timing. 83% of respondents indicated that they previously hadn’t used an automated system to manage the racing.

Chart, pie chart

Description automatically generated

The results from Q2 show that 2/3rds of the respondents indicated they prefer using a software option rather than working out the results manually, this is perhaps not unexpected as the users that indicated this tended to be more tech-literate. The users who prefer the software option also tend to perform the role more so may find using an automated process has an increased efficiency.

Chart, pie chart

Description automatically generated

### Existing Software Analysis

#### Introduction and overview

In the market currently there are 2 different types of solutions available, these are:

* Solution 1 - A management and timing system of the race itself, often features functionality to add racers, start and finish the race and display a summary of a results sheet.
* Solution 2 – A records keeping system for races, often can manage a series of races. Usually, database style features are available as well as customisation features to allow individual clubs to setup race schedules.

#### Currently available soloutions

##### HalSail

The most popular app for solution 2 is called HalSail (Hal), this is the app that my local sailing club uses for storing the races. Hal works solely based on user input, this includes the start and finish time of each boat. There are no timing features within Hal. Hal maintains a database of different sailors and their classes and allows the race officer to add each to a race. The main feature of Hal is a deep feature set of historical races and allowed computation of multiple series.

Positives:

* Very in-depth back end allowing an administrator to execute queries for series podium prizes.
* Contains a repository for storing boats and their respective sailors to allow simple adding to races.
* Has the ability to customise handicap values (PY) including adding new craft.
* Provides clients with a results sheet which can be shared online easily.
* Allows the input of special race classifications e.g. OCS – Over the line

Drawbacks:

* Although feature rich the user interface displays this badly, this often causes difficulties for those not used to the application.
* Although a craft may finish the race in hours and minutes e.g. 1 hour 10 minutes, the program only accepts a finish time in 24 hr format. This means for each boat a finish time must be calculated by adding duration to the start time.
* There is no start time functionality, this means the 5-minute sequence must be managed separately.
* The ability to add sailors who sail multiple classes of boat is poor and often causes confusion.

##### Sail Results

An app which I used previously was called sail results. Although this app is no longer functioning on the latest version of android it was intended to solve some of the problems from solution 1. This app provided an interface which started with a countdown timer and allowed boat classes to be added to the race, it was then up to the user to ensure all craft were added and then start the race along with lapping. At the end of this race the app would display results however its long-term storage functionality was very limited.

Positives:

* Used a sailing timer to start the race, providing sounds at each interval (intervals are a common standard for sailing watches).
* Allowed the restarting of the countdown with ease
* Allowed the boats to be added prior to the countdown starting
* Provided the ability to lap boats
* Had a good summary of results on the final page including the ability to email results

Drawbacks:

* The UI was not user friendly which led to the following problems
  + The functionality to add a boat to the race relied on the user entering a CSV style list with no spaces, this was unhelpful for non-tech savvy officers.
  + There was no button to finish craft, instead the last lap function was used. This often-caused confusion.
  + It was particularly difficult to set a schedule of races, this caused issues where multiple races could accidentally be added.
* There were no collections of boats that had sailed previously, this meant the app relied on the race officer individually adding each sailor and boat for each race or series. This method was prone to mistakes and there was no functionality to rectify.
* Customisable PY (Handicaps) could not be added, this meant some classes without predefined handicaps could not be raced.
* There was no ability to add special result classifications e.g. OCS – Over the line.

##### SailWave

SailWave is an application which doesn’t directly solve either of the solutions but regardless is an important application to look at due to its widespread use in many clubs and regattas.

SailWave is a much simpler version of both of the above applications, its design is based around all of the racing craft being the same fleet with the same PY. It can also be used for handicap racing but this is not the main feature of the application. SailWave can be likened to an excel spreadsheet in its management of results and is mainly used by professional race officers and larger racing clubs. This is possible because many of the race organisation teams do the manual recording on paper prior to adding to SailWave.

As SailWave isn’t designed to implement either solution above we can’t draw positives and drawbacks however there are still takeaways from what it does well. This includes:

* A simple administration interface, this is particularly useful for series customisations.
* Interface can be likened to the results printout; this provides instant feedback for the race officer.
* Boats can be added very quickly to the races.

#### Takeaways

The main aim of my project is to combine solution 1 and 2 into one product, this means its key to retrieve takeaways from existing products that implement or partially implement each. I should also retrieve takeaways from other similar solutions such as the SailWave application.

The takeaways I aim to include are shown below, this list only includes possible component improvements from the analysed solutions. These should form part of the requirements refinement process:

* There should be a query results function to allow an administrator to select the results they wish to view.
* There should be a store of boats matched up to sailors allowing race officers to add craft to races easily.
* It should provide PY handicap valuation manipulation with stock handicap values being gained from online services.
* Provide a sharable link for results so racers can easily see how they performed.
* Allow special classifications to be added instead of numerical place values.
* Implement a timing solution that functions based on a stopwatch mechanism.
* Be able to lap and finish boats individually as they cross the line.
* A customisable starting sequence should be implemented including a sailing style timer.
* A series setting ability should be available to set when races should start.

### Requirements

A total of 45 requirements were created from the various requirement gathering techniques. This includes 37 functional requirements and 8 non-functional requirements. Each of the requirements has an ID, a description, a rationale indicating why the require is needed, and a fit-criterion to indicate when the requirement has been met.

A full list of the requirements can be found in the appendix

### Prioritisation

Two requirement prioritisation methods were used to set the priority levels of each of the requirements. These were MOSCOW prioritisation and relative weighting. Once prioritisation was complete the MOSCOW levels were added to the requirements table.

#### Moscow

For this project the MOSCOW requirements process was used to provide the stakeholders with an immediate reference that they could use to indicate if they were happy with which items must be included, should be included, could be included, or won’t be included. At this point the main stakeholder had the opportunity to make changes to the priority level where applicable. The requirements priority from a MSOCOW context can be seen in the final column of appendix B.

Graphical user interface, text, application

Description automatically generated

#### Relative weighting

Relative weighting was also used throughout requirements prioritisation stage to provide granularity including which features may be quick wins and which many require more effort. This technique was used to provide more information throughout planning and development rather than as an indicator for the stakeholder. A table was produced and indicated the weightings using relative benefit, relative penalty, relative cost, and relative risk.

Table

Description automatically generated

## Risks

A extensive risk table was created to show the calculated risk impact for each of the requirements. These were calculated using a severity index and likelihood impact. The calculation of these risks assists with the planning for each of the work packages e.g. if an item includes more risk it may be correct to include it in an earlier work package. An example of the risk table is below, the full risk assessment can be found in appendix B. 1 indicates a low score with 5 indicating the maximum.

Text, table

Description automatically generated

The risks were also transposed into a risk matrix where a visual representation of each of the risks are show. It shows the distribution of the risks before and after mitigation. There is a distinct reduction in the risks at higher levels.

A picture containing table

Description automatically generatedA picture containing table

Description automatically generated

# Project Planning

## Work breakdown Structure

The structure for this project has been subdivided into 6 headings. This would normally be 5 but as this is a final year project an extra heading has been added for final documentation and presentation preparation. Each of the headings has been subdivided to show each of the tasks that need to be completed for the project to be a success. More information can be found about Work Package execution later in the document.

A screenshot of a computer

Description automatically generated with low confidence

Once the WBS had been decided each of the tasks had to be estimated. This was carried out using PERT estimation. Estimates in number of days for optimistic, likely, and pessimistic calculate to produce. PERT value which when rounded to the nearest 0.5 produces an estimate for that task in man days. An example of this is shown below.

Graphical user interface, application, table

Description automatically generated

These estimates and tasks will form the basis of a Gantt chart which is shown in the next section.

## Gantt chart

Timeline

Description automatically generated with medium confidenceChart

Description automatically generated

The creation of the Gantt chart centerer around the items in the work break down structure being given PERT estimation values. This forms the WBS structure seen on the left. In addition, milestones have been added. This has allowed deadlines to be set e.g., project completion.

Each item in sequence could then be added to the chart with items that needed to be completed before others marked as predecessors. This provided accurate projected sequencing.

A major part of the Gantt chart was the work packages. Further information about this can be found in the project methodology section. In this project I decided that 3 work packages would fit the requirement sets best.

## Chosen Methodology

The methodology I’ve chosen for this project a variation of the Kanban Agile methodology. This methodology allows a continuous way of working and doesn’t rely on segmenting the work into short sprints which is often time consuming. Instead, a large backlog can be created using cards where when a card is being worked on it is moved into appropriate columns. This means the capacity is equal to the maximum capacity at any time. I plan to create 3 work packages to separate the work into features and the MVP.

This methodology will involve breaking the requirements into actionable tiles to and adding them to the backlog. These tiles can them be worked on in a priority order completing each feature individually before picking up the next one.

### Reasoning

#### Responsibilities

One of the main reasons for choosing Kanban is the ability to work on many of the features easily with autonomy. Unlike other agile methodologies such as scrum where a project manager is required to allocate tickets to sprint, Kanban allows any developer to pick a ticket from the backlog and perform a technical refinement to bring it into development. As I am working on this deliverable alone I can organise all tickets in backlog by myself.

#### Deliverys

All features and products in the delivery are produced continuously on an as needed basis. This is the ‘just in time’ concept. Features are often assigned a priority level from which a developer will pick up a task in order, this means the items for MVP will be completed first. This helps my project as I can work flexibly on tasks without having set deadlines for tickets as long as they are completed by the project deadlines.

#### Code Management

As I am working alone, I will be the only one actively using source control. This means that when I complete a feature, I can merge the code into the main branch in a chronological order. This will simplify my way of working as with Kanban I won’t be working on multiple features at once.

#### Modifications and Changes

The Kanban methodology allows for changes to be made mid-way through the development cycle. This means different iterations and continuous improvements can be made as the tasks are set chronologically. This is something that is harder to do in a scrum methodology where the tasks are set pre-sprint start.

#### Adaptability

Adaptability is key for projects that require ongoing changes to be made to key pieces of work. This allows this work to be performed without having to revisit previous sprints such as would occur in scrum methodologies.

#### Understandability

A key advantage to using the Kanban methodology is that management tasks are simplified due to limiting the work in progress to the capacity. This means focus can be allocated to the feature in hand without any distractions from other tasks. Features are often understood better through in-depth working.

### Work Packages

In my project I’ve divided my requirements into 3 work packages. Each of these work packages form a different function or capability. This is key to reducing risk around my project and improving tracking from a management perspective. Each of these work packages will function in a Kanban style fashion and will be iterative until the work package is completed.

#### Work package 1

This work package will contain the full minimum viable product. It should contain all the main required features that make up the system from the perspective of a race officer including timing and a results printout. The areas not covered by the MVP include data storage for series and races as well as any configurable items and settings.

This is projected to be the largest MVP as a large amount of the application setup is performed here. This is also the part of the project with the largest risk. Completing it first will allow the risks to me controlled and extra time assigned where necessary.

#### Work Package 2

Work package 2 will contain the majority of the data facilitation features such as anywhere large amounts of intricate data bust be stored for future use. This work package will touch a lot of the work previously completed in work package 1 as well as creating a number of data manipulation features for the administrators including the creation of races and series.

#### Work package 3

This final work package will hold the configuration and options features. This includes a large proportion of the edits the administrator will be able to make. Within this is ability to customise and add boats, sailors amongst others.

Text

Description automatically generated

## Technology Analysis

Based on the requirements it is immediately clear that the solution should be based around portability. This gives the indication that the technology chosen should be based fundamentally around an app or web app. There are multiple different technologies with use cases for each. In this section I will look at the advantages and disadvantages of implementing each and apply them to the requirements for the project.

### Xamarin

Xamarin is a .NET based developer platform focused on adaptability and the use of a single code base across multiple platforms. Development can be carried out within visual studio on both windows and OSX.

* Advantages
  + Very good cross platform support with minimal changes to individual OS code bases allowing the app to run efficiently on both iOS and Android
  + Development uses C♯, a widely known and supported language amongst developers and projects.
  + Simple hot plug developer debugging on live devices.
  + Developer working on the project is versed in Xamarin.
  + Integrations with Azure provide simple serverless functionality.
  + Faster development process.
* Disadvantages
  + Not as efficient as native apps.
  + Hard to use with large amounts of platform specific functionality.
  + Not appropriate for apps that deal with a large amount of data locally.

### Android Studio

Another platform that was considered is Android Studio. This is a native android development platform based on the Kotlin language. Development can also be carried out in iOS or OSX.

* Advantages
  + Ideal app performance as the app will be native to the platform.
  + A much more customisable interface is available when compared to cross platform solutions.
  + Suited for apps with large quantities of data.
  + Implementation of native behaviour is better.
* Disadvantages
  + App can only run-on android.
  + Kotlin has a steep learning curve.
  + Slower development process.
  + UI can only be used within the android ecosystem.

### Choice

For this project Xamarin was chosen to be the technology used. Xamarin’s use of .NET for cross compatibility is a key benefit. Unlike android studio the code base will also work with iOS, something that would be a key advantage where a lot of sailing organisations already have iPad devices. Reducing complexity in this area rather than deploying two code bases is a significant advantage.

Xamarin is also a technology that is very familiar to the developer having worked extensively with it in the past. This is a key advantage over learning Kotlin and Android Studio.

Although an advantage to native android is the ability to edit native behaviour and device level actions this is not anticipated to be needed throughout the project. There is a much greater advantage to a simple cross-platform solution.

# Design

# Appendix

# Appendix B

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Description | Rationale | Fit Criterion | Priority Level |
| FR1 | A list of craft and their owners should be visible to allow users to add boats to races. | Craft should be selectable by the race officer to include them in any race. This prevents repeat entry of information. | A race officer should be able to select the list of racing craft and add them into a race. | M |
| FR2 | Administrators should be able to add craft to the boat list including the owner’s name. | If new racers or boats become involved, they can be added to the system. | Administrators can add a new craft to the boat list providing all the necessary fields. | M |
| FR3 | The race officer should be able to edit the crew for any race that boat is part of. | The crew of a craft can change between each race, the crew should be included in any results so should be editable. | The race officer can set the crew’s name pre or post-race. | C |
| FR4 | Administrators should be able to set the craft PY if required to alter from PY list. | On occasion the PY of a craft should be edited. This may be because of a crew number or weight change. PY’s, although unlikely may also be corrected mid series. | A new PY value for any craft class can be set. | C |
| FR5 | Administrators should be able to import a new PY list. | PY lists change between years, a new PY list should be importable. | The csv PY list can be imported to the application and the changes will take effect from that point forward. | M |
| FR5 | Administrators should be able to remove craft from the boat list. | Any craft or helm that isn’t active or is no longer a member should be removable from the boat list. This will cut down on unnecessary searching for a craft. | Administrators can delete craft; these craft should remain in races already completed but should not be available for racing going forwards. | M |
| FR6 | The user should be able to set the race officers name and safety crew names. | Recording who the race officer and safety crew is important for post-race analysis and who should be asked for any questions. | The race officer and safety crew names can be set, this also should be available for additional duties. | S |
| FR7 | The user should be able to assign redress for the race officer and safety crew. | If someone performs a duty, they are given redress. This means the race officer should be able to provide redress to those on duty. | Redress values are automatically assigned for those on duty. | S |
| FR8 | The current weather conditions should be retrieved and stored in the app for future reference. | Recording the weather is important when manually validating results. If the conditions are in the extreme (very light or very windy) results may be extreme. Having a record of the conditions are important to verify this. | Weather conditions are automatically retrieved from external sources and shown in the app. These values can be edited. | C |
| FR9 | The app should have the ability to schedule the races for a date, these races should only be displayed on the current date. | To avoid extra steps administrators should be able to schedule races so the race officer only must select the race to start it. This is important to make the race officers job as easy as possible. | A race officer can select and start a race without any configuration. | M |
| FR10 | The app should have the ability to schedule series and include races in those series. | Most races should be part of a series, the series are used to show how racers have performed throughout a season or part of a season. This is also important for events such as regattas. | Series should be scheduled, and races can be included in the series scheduled. The races perform the same as races outside of a series. | M |
| FR11 | The user should be able to select a race that is about to begin, this should be one of the races displayed. | Race officers should be able to start the races on that day. | The race officer can start any races scheduled for the day. | M |
| FR12 | The user should be able to start the race countdown via a start button. | Countdowns should be available to start at any time to allow flexibility in the timing. Race officers often must multitask, so this is important. | Even though the scheduled race time has not arrived or has passed the race officer can still start the race. | M |
| FR13 | The countdown should be a sailing style i.e., has intervals at 5,4,1,0. | Sailing countdowns are specific, to assist the race officer with their signal timing the app should provide prompts at intervals. | The intervals and total time can be configured as appropriate. | S |
| FR14 | The countdown should be able to be reset at any time. | If the race needs to be restarted e.g., for a general recall there should be a reset time function | The race officer can restart the race at any time. This includes being able to reset the starting sequence. | M |
| FR15 | The app should have the ability to record boats that are OCS or BFD (over the line or disqualified due to starting errors). | Special results assignments should be provided and can be assigned to a user. There are many classifications that should be available to be added. | The race officer can set special classifications for any craft at any point during the race. | S |
| FR16 | Upon race start, a stopwatch should be started recording the length of the race. | A recording of how long the race took should be readily available for reference both during and post-race. | A visual representation of the time is available throughout the race and after. | M |
| FR17 | A screen showing the racing status of each boat should be visible, this should contain options. | The options should include finish and lap options as dictated in other requirements. There should be as many boats listed as possible on that screen. | A minimum of 10 boats should be shown on the screen at any time throughout the race. | M |
| FR18 | The user should be able to ‘lap’ each boat, the finish times should be defaulted to the lap times should the race be shortened. | Lap times are important so that in the eventually that a race is shortened the race results can be calculated based on the lap previous. | Lap time options for each boat are usable. When the race is shortened, the results are automatically displayed based on the last lap. | M |
| FR19 | The ability to finish each boat with a single button press should be available. | Craft should be as easy to finish upon line cross as possible. This is important as multiple craft can finish in close succession. | The race officer can finish 5 boats within 2 seconds. | S |
| FR20 | There should be a finish all button that takes the last lap time of each boat | If a race is shortened all boats should be finished based on the previous lap. | The finish all button calculates the results based on the last lap. | S |
| FR21 | Boats that will be timed out should be placed based on their order on the course. This should be configurable. | Some clubs place boats that are timed out. This means the race officer should be able to assign places to those boats. | The timed-out boats can be allocated a place if appropriate. | C |
| FR22 | Once all boats have completed the race the application should show the results sheet. | The results sheet should be presented to the race officer to display in the club for racers to view how they performed. | The results screen is shown immediately after race completion in a simple readable format | M |
| FR23 | The results screen should show all the craft with their place information. | Manual verification is important to check there are no anomalies. | The results screen shows timing allowing for manual verification. | M |
| FR24 | The adjusted results should be calculated via an up-to-date PY list provided by the RYA | Having an up-to-date PY list keeps the racing on a fair level. Each year the list is adjusted to improve the fairness, in particular of new classes of craft. | The latest PY list for handicaps is used. | M |
| FR25 | The results should be editable should there be any post-race changes e.g., Protests. | Having a list of editable results is important if there are any disputes or manual changes that need to be made for validity of races. | All result times and points are editable post-race at any stage by the administrator or the race officer on that day. | S |
| FR26 | The results file should be stored to allow later viewing. | Results should be viewable at any point for both the race officer on duty as well as the chief race officer for validation and display. | Anyone can view the results without a password at any time. | M |
| FR27 | If appropriate the results file should be stored as part of a series accessible by user and admin. | If a race is scheduled as part of a series, it is logical it should be included as part of the series results. | A user can view the overall series results as well as each race contained inside. | M |
| FR28 | The series results should be calculated according to each of the sub-races. | For each series, each of its races should contribute to the overall result. This is how series winners are generated. | Overall results are calculated automatically for each series. | M |
| FR29 | Series manipulation should include scoring discards. | Adding options for changing how discards are processed is important as there are different discard schemes depending on the event or club. | Discard schemes can be altered to different definition patterns e.g., 1 discard after 6 races | S |
| FR30 | Series manipulation should include setting specific special values | Special classifications are important for many scenarios where a normal result isn’t appropriate. Many clubs set their own special values so these should be configurable. | The administrator can set values for all special values. | C |
| FR31 | Special scoring values should be available to be set for special redress results. | Redresses can be provided for a number of reasons, the administrator should have the option to assign these where they see fit and to any value. | The administrator can set redress to any craft for any race at any time. | C |
| FR32 | Race officers should have the ability to print summary records. | Printing ability is important as defined by the international racing rules. This is so any results can be displayed on an official notice board. | Race officers can print any of the races and results sheets. | M |
| FR33 | The ability to customise the starting sequence should be available. | Clubs often have different starting sequences e.g., 3-minute countdown instead of 5. There should be an option to set this. | The administrator can set a special starting sequence. | C |
| FR34 | Any administrative areas should be protected via a configurable administrative password. | To prevent unauthorised changes administrative areas should be protected. | The race officers and other users are unable to access elevated areas without using a password. | M |
| FR35 | All users should be able to execute results queries, this may be in relation to the progress of racers in a series. | Racers often wish to view how they are performing in the standings throughout a series. This should be available for all to see at any stage. | Users can search the series and races to view how they are performing at any point. | S |
| FR36 | Analysis tools should include being able to draw turnout metrics. | Often at the end of the season a racing committee will evaluate the performance of racing and the dates on which races were run. Metrics describing turnout are useful for this. | The administrator can view the turnout for races and series. | C |
| FR37 | Analysis tools should include being able to calculate the time gap over a 1-hour period for pursuit style races. | Sometimes clubs run a ‘special’ pursuit race. This is where the PY times over an hour are adjusted and the race turns into who can cross the line first rather than adjusting for time. | A timing sheet can be generated showing the start times for each class of boat. | C |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Description | Rationale | Fit Criterion | Priority Level |
| NFR1 | The app must be displayed on a large screen device such as a tablet. | It will be difficult to display the output of information required on a small screen device so may users will prefer a larger screen. | The app scales to a 13-inch display. | S |
| NFR2 | The app should be functional without a network connection, any network activity should be performed when re-connected. | Often the race officer will not be in connection with either WIFI or mobile data as they are in a boat on the water. Provisions should be made for this. | The core functionality of the app should be unaffected by lack of connection. | M |
| NFR3 | The app should have the minimal amount of button presses to start and stop the race. | The race officer may need to perform many actions outside of using the app. This app should be as simple as possible to cater for this. | The race officer should be able to start a race within 5 button presses. | S |
| NFR4 | The app must display as many boat entries as possible on the race screen. | There may be cases when between 3-5 boats cross the line within a few seconds. The race officer should be able to finish each accurately and not spend time navigating. | The app can display at minimum 10 boats on a 12-inch tablet. | S |
| NFR5 | The app must support the latest OS version. | Where supported by the manufacturer the app will run on the latest version of the operating system and be backward compatible. This ensures the largest number of devices are supported. | The app runs on the latest version of the chosen OS. | M |
| NFR6 | Data should be stored locally. | This will allow access and storing of data where there is little signal. This is important to be able to recall race data. | Past results can be retrieved without an internet connection. | C |
| NFR7 | The performance of the applications should be very responsive. | This is important to record correct times for finishing boats. | The app is overall performant. | M |
| NFR8 | The application should conform to the World Sailing RRS (Racing Rules of Sailing). | Conformation to the rules is important to ensure there are no gaps in the application which may cause rule issues. | The app has no errors that may result in protests (as judged by a Race Juror). | S |

# Appendix C

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Risk Id | Description | Consequence | Mitigation | Severity Index | Likelihood Index | Risk Impact |
| A | The overall time for project completion may be miscalculated. | This could result in the project or parts of the project not working before the final review. | The requirements should be broken into phases where each work package encompasses a feature. This is likely to mean WP1 is equal to an MVP. Should the project not be fully completed there should be 2 fully functional packages. | 4 | 3 | 2.4 |
| B | The scale of the application development could be underestimated. | This could result in the development progress being slower than expected. | Requirements should be made as accurately as possible. They should encompass the entire product minimising any gaps. This should improve the estimation capability. | 3 | 1 | 0.6 |
| C | There may be bugs present in prior work packages that lead to investigation and fixing. | This will extend the development time of upcoming work. | Development testing should be a top priority to catch issues early. Testing should be performed throughout the project with both manual and automated tests implemented. | 1 | 3 | 0.6 |
| D | Task complexity can be underestimated. | Individual tasks may take longer than projected leading to feature delays. | Extensive architecture and technical designs should be produced early to ensure there is a sound technical foundation to each component. | 2 | 4 | 1.6 |
| E | There could be changes to the requirements specification within the project development. | Some work that has already been completed may need to be reviewed and altered. | All requirements should be thoroughly examined with all involved stakeholders to ensure they are set appropriately. For any changes they should be made through the set change control procedure to ensure minimal impact. | 4 | 2 | 1.6 |
| F | Requirements may have been constructed incorrectly resulting in misinterpretation | This could result in effected features being developed incorrectly and fail to achieve their intentions. | All requirements should be thoroughly discussed with the stakeholders and that everything makes sense from a development perspective to reduce misinterpretation. | 3 | 2 | 1.2 |
| G | Requirements aren’t correctly split into packages and dependencies are formed leading to MVP not being completed. | The features may not be developed in segments increasing the risk that different components may not be completed. | Prioritisation techniques should be implemented to ensure the MVP contains all the significant requirements. All of the requirements should be matched with any dependencies to ensure they are developed in the correct order. | 2 | 2 | 0.8 |
| H | There are not enough requirements meaning assumptions have to be made throughout development. | If requirements have left gaps there may be issues where tasks aren’t fully defined. This can sometimes result in wasted effort. | Clarification on all tasks should be gained with the stakeholders. | 2 | 1 | 0.4 |
| I | Data failure may occur or cause issues. | This could mean parts of the project are lost due to corruption, misconfiguration, infrastructure issues or physical hardware damage. | Backups should be made effectively including the use of multiple offsite backups. All development work should be contained within source control methods. If data is lost set recovery methods should be actioned. | 5 | 1 | 1 |
| J | Framework or platform issues may occur resulting in issues within the application. | Major platforms updates such as a new Android version can cause issues within the project including changes that need to be made to ensure compatibility. | Throughout the development attention should be paid to release schedules for any platforms or frameworks that should be used. If possible, access to beta builds should be gained to ensure as much time is allowed for changes to be made. | 4 | 3 | 2.4 |
| K | There may be security concerns or vulnerabilities uncovered. | Security issues may need fixes to be applied or updates implemented to ensure security. | The app should be made with minimal reliance on external libraries and software. This means that any security issues will be limited and should be controllable. | 3 | 2 | 1.2 |
| L | Code debt or longevity issues may be uncovered within the later stages of the lifecycle | The app could lose functionality or future tasks may be harder to develop. | Technical refinements should be in-depth to ensure the development solution is the most practical for the longevity of the project. Proven technologies should also be used where possible. | 2 | 3 | 1.2 |
| M | The chosen tools may not be the most effective for the tasks. | The development time and efficiency may be negatively impacted. | A review of the chosen technologies should be performed beforehand to evaluate which solutions may be best suited. | 1 | 2 | 0.4 |

# Appendix C

Graphical user interface, application, table

Description automatically generatedTable

Description automatically generatedGraphical user interface, application, table

Description automatically generated

Table

Description automatically generatedTable

Description automatically generated

Table

Description automatically generated