

National College of Ireland

BSc in Computing

2016/2017

Niall Quinn

X13018727

niall.quinn1@student.ncirl.ie

RFBBase

Technical Report



Declaration Cover Sheet for Project Submission

SECTION 1 *Student to complete*

Name: Niall Quinn
Student ID: X13108727
Supervisor: Padraig DeBurca

SECTION 2 **Confirmation of Authorship**

The acceptance of your work is subject to your signature on the following declaration:
I confirm that I have read the College statement on plagiarism (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature: _____ Date: _____

NB. If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the College's Disciplinary Committee. Should the Committee be satisfied that plagiarism has occurred this is likely to lead to your failing the module and possibly to your being suspended or expelled from college.

Complete the sections above and attach it to the front of one of the copies of your assignment,

What constitutes plagiarism or cheating?

The following is extracted from the college's formal statement on plagiarism as quoted in the Student Handbooks. References to "assignments" should be taken to include any piece of work submitted for assessment.

Paraphrasing refers to taking the ideas, words or work of another, putting it into your own words and crediting the source. This is acceptable academic practice provided you ensure that credit is given to the author. Plagiarism refers to copying the ideas and work of another and misrepresenting it as your own. This is completely unacceptable and is prohibited in all academic institutions. It is a serious offence and may result in a fail grade and/or disciplinary action. All sources that you use in your writing must be acknowledged and included in the reference or bibliography section. If a particular piece of writing proves difficult to paraphrase, or you want to include it in its original form, it must be enclosed in quotation marks

and credit given to the author.

When referring to the work of another author within the text of your project you must give the author's surname and the date the work was published. Full details for each source must then be given in the bibliography at the end of the project

Penalties for Plagiarism

If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the college's Disciplinary Committee. Where the Disciplinary Committee makes a finding that there has been plagiarism, the Disciplinary Committee may recommend

- that a student's marks shall be reduced
- that the student be deemed not to have passed the assignment
- that other forms of assessment undertaken in that academic year by the same student be declared void
- that other examinations sat by the same student at the same sitting be declared void

Further penalties are also possible including

- suspending a student college for a specified time,
- expelling a student from college,
- prohibiting a student from sitting any examination or assessment.,
- the imposition of a fine and

the requirement that a student to attend additional or other lectures or courses or undertake additional academic work.

Table of Contents

1 - Executive Summary	7
2 - Introduction	7
2.1 - Background	7
2.2 - Project Scope	8
2.3 - Technical Approach	8
2.4 - Technologies	9
2.4.1 - Backend Platform	9
2.4.2 - API	9
2.4.3 - Database	9
2.4.4 - iOS Application	9
2.4.5 - Android Application	9
2.4.5 - Technology Overview	10
2.5 - Resources	10
2.5.1 - Literature	10
2.5.2 - Tooling	10
2.5.3 - Hardware	11
2.6 - Definitions, Abbreviations & Acronyms	11
1.3 Definitions, Acronyms, and Abbreviations	11
3 - System	12
3.1 - Requirements	12
3.1.1 - User Requirements	12
3.1.2 - Functional Requirements	13
3.1.2.1 - Overview	13
3.1.2.1.1 - Admin Platform Functional Requirements	13
3.1.2.1.2 - API Functional Requirements	13
3.1.2.1.3 - Mobile Application Functional Requirements	14
3.1.2.2 - ADM-1 - User Authentication	14
3.1.2.3 - ADM-2 - Customer Creation	15
3.1.2.4 - ADM-3 - Customer Termination	15
3.1.2.5 - ADM-4 - News Creation	16
3.1.2.6 - ADM-5 - News Item Editing	17
3.1.2.7 - ADM-6 - Delete News Item	17
3.1.2.8 - ADM-7 - Add Social Stream Details	18
3.1.2.9 - ADM-8 - Edit Social Streams	19
3.1.2.10 - ADM-9 - Add Media Items	19
3.1.2.11 - ADM-10 - Remove Media Items	20
3.1.2.12 - ADM-11 - Broadcast Notifications	21
3.1.2.13 - ADM-12 - Edit App Theme	21
3.1.4 - Non Functional Requirements	22
3.1.4.1 Performance/Response time requirement	22
3.1.4.2 Availability requirement	22

3.1.4.3 Recover requirement	22
3.1.4.4 Security requirement	22
3.1.4.5 Reliability requirement	22
3.1.4.6 Maintainability requirement	22
3.1.4.7 Portability requirement	23
3.1.4.8 Extendibility requirement	23
3.1.4.9 Reusability requirement	23
3.1.4.10 Database Requirement	23
3.1.5 - Graphical User Interface Requirements	23
3.1.5.1 - Interface Elements	23
3.1.5.2 - WYSIWYG Editor	23
3.1.5.3 - Color Picker	23
3.1.5.4 - Modal Views	23
3.2 - Use Case Diagrams	24
3.2.1 - Admin Platform Use Case Diagram	24
3.2.2 - API Use Case Diagram	24
3.2.3 - Mobile Application Use Case Diagram	25
3.3 - Implementation	25
3.3.1 - Backend Platform	25
3.3.2 - Mobile Applications	26
3.3.3 - REST API	26
3.3.4 - Theme Engine	26
3.3.5 - Broadcast	27
3.3.6 - White Labeled Apps	27
3.3.7 - Deployment	27
3.3.8 - Security	27
3.4 - Functionality	28
3.4.1 - News	28
3.4.2 - Bio	28
3.4.3 - Calendar	28
3.4.4 - Theme	28
4. Testing	29
4.1 - Usability Testing	29
4.1.1 - Trunk Test	29
4.1.2 - 5 Second Test	31
4.1.3 - Prototyping	31
4.2 - Unit Testing	33
4.3 - Integration Testing	34
4.4 - Graphical User Interface	37
4.4.1 - Admin Platform GUI Mockup	37
4.4.1.1 - News Page	37
4.4.1.2 - Social Page	38
4.4.1.3 - Broadcast Mockup	38
4.4.1.4 - Gallery Mockup	39

4.4.1.6 - Theme Page Mockup	40
4.5 - Application Programmers Interface (API)	40
4.5.1 - API Specification	40
4.5.2 - API Sequence Diagram	42
4.6 - Architecture	43
4.6.1 System Class Diagram	43
5 - Project Plan	43
5.1 Gantt Chart	44
6 - System Evolution	44
7 - Useful Links	44
8 - References	45
Appendix 1 - Project Proposal	46
1. Executive Summary	47
2. Background	47
3. Technical Approach	48
3.1 Technical Overview	48
4. Resources	49
4.1 Literature	49
4.2 Tooling	49
4.3 Hardware	49
5. Project Plan	50
5.1 Gantt Chart	50
6. Technical Details	50
6.1 - Backend Platform	50
6.2 - API	50
6.3 - Database	50
6.4 - iOS Application	51
6.5 - Android Application	51
7. Evaluation	51
7.1 - Unit Testing	51
7.2 - Manual Testing	51
7.3 - Requirements Evaluation	51
Appendix 2 - Monthly Journals	52
Appendix 3 - Usability Questionnaire	57
Appendix 4 - Showcase Poster	59

1 - Executive Summary

The aim of this project is to produce a highly functional marketing platform for racing drivers. The platform provides valuable functionality to the customer, and act as a catalyst to grow their social media following. The project will be provided using modern languages, frameworks and tooling. This platform includes a custom theme engine, which will allow the mobile application's look and feel to be configured from the backend platform. The platform employs multi-tenancy on the backend - data is segmented by customer and surfaced to the customer's personal mobile applications on the front end. The platform will be developed with security in mind. The data will be kept safe and confidential, and there will be no overspill of data between tenants.

RFBBase is a marketing platform for modern race drivers. Gaining the marketing edge is more important than ever, with drivers fighting for sponsorship and race drives, getting the word out is a top priority. RFBBase gives them a platform to deliver News, Driver Bio, Race Calendar, Social Streams and Push Notifications to their fanbase. I achieve this by supplying the driver with a login to the platform where they can generate all of their content. They will also get two white-labeled mobile applications, deployed to the iOS App Store and the Google Play Store. Drivers can take residency on the home screens of their most loyal fans, and use RFBBase to deliver quality, exclusive content.

Upon evaluation I found that this product does address the needs of many racing drivers. Drivers are very excited at the prospect of having their own app in the app store, and see a benefit to having this personal stream to broadcast to their fanbase. The theming engine was a big hit among all users - the ability to customise the look and feel of the applications on the fly being identified as a big selling point.

2 - Introduction

2.1 - Background

Motor racing drivers today spend a large amount of time marketing their brand to the large motor racing fan base. At the moment, drivers use traditional social media, Twitter, Facebook, Instagram etc as a means to reach their audience. There is a place in the market for a single outlet from which the driver can target and grow their audience.

A customer on the platform will receive a login to their own backend portal, where they will be able to add various information such as: News, Race Calendar, Images & Video and Social streams. All of this information will be served to their own custom-themed iOS and Android apps which will be deployed via the iOS App Store and Google Play Store respectively.

The business model for this product is a SAAS model. A Customer on the platform will pay an upfront development fee, and then pay a monthly subscription to use the service.

I see a gap in the market for this product, and am confident that when deployed will be a viable product.

2.2 - Project Scope

The scope of the project is to develop the entire system which will be designed from the beginning to work in harmony. The system consists of an application which will encompass the API, an application which will encompass the admin backend and front-end marketing site, and mobile applications for the iOS and Android platforms

Various racing drivers were consulted during the product planning phase, which has led to the below requirements being laid out.

2.3 - Technical Approach

This project was developed using modern languages, frameworks, tooling and infrastructure. A clear and concise project plan was set out in order to maximise time available and ensure all parts are developed in time, and integrate seamlessly. At regular intervals, progress was assessed and the project timeline updated accordingly.

From a high level, this project is comprised of five main components:

1. Backend portal - a Ruby on Rails Application
2. API - a Ruby on Rails API
3. Database - a PostgreSQL database using Heroku Postgres
4. iOS Application - hybrid application developed using Swift and Turbolinks [1]
5. Android Application - hybrid application developed using Java and Turbolinks

I endeavoured to identify all requirements in the requirements specification stage, in order to build the platform from the beginning with all entities and relationships in scope. This led to less headaches in the testing and integration phases later.

The project employs a continuous integration scheme, with automated unit testing. All code merges to the master branch will pass through the automated testing framework, with bad builds being rejected. This ensures that critical bugs do not make it to the production code.

The backend platform makes use of the Heroku deployment toolchain. This allows me to choose one of Heroku's many price points to deliver my platform, and also to utilise the excellent deployment tools provided by Heroku. If the load on the app or API becomes too large, Elastic Beanstalk will automatically create new instances of the application to handle this load. This provides us with an extra layer of robustness to scaling problems.

A key part of this product will be the custom theming engine for mobile. The mobile applications pull down a specially formatted JSON file from the API. Acting upon this file, the application will initialize core colours, fonts and images. This results in each of the platform customers receiving a customized application, to suit their own brand.

The latest technologies in automated building is used on the mobile apps, to reduce developer input when onboarding new customers. The *Fastlane* platform is used to execute custom scripts which will take care of app building and distribution. This is a key area to focus on as the business scales in order for customer onboarding to not become a headache.

Requirements were captured by interviewing current racing drivers. I asked them what they would like to see in the application and build up a ranking of features from those interviews.

2.4 - Technologies

2.4.1 - Backend Platform

Implementation Language	Ruby, HTML (HamI), CSS (SCSS), Javascript
Deployment	Heroku Dyno
Libraries Used	Ruby on Rails, Various RubyGems as per requirements, Bootstrap, RSpec test framework

2.4.2 - API

Implementation Language	Ruby, serving HTML & JSON over REST interface
Deployment	Heroku Dyno
Libraries Used	Ruby on Rails, Various RubyGems as per requirements, RSpec test framework

2.4.3 - Database

Implementation Language	PostgreSQL
Deployment	Heroku Postgres
Libraries Used	

2.4.4 - iOS Application

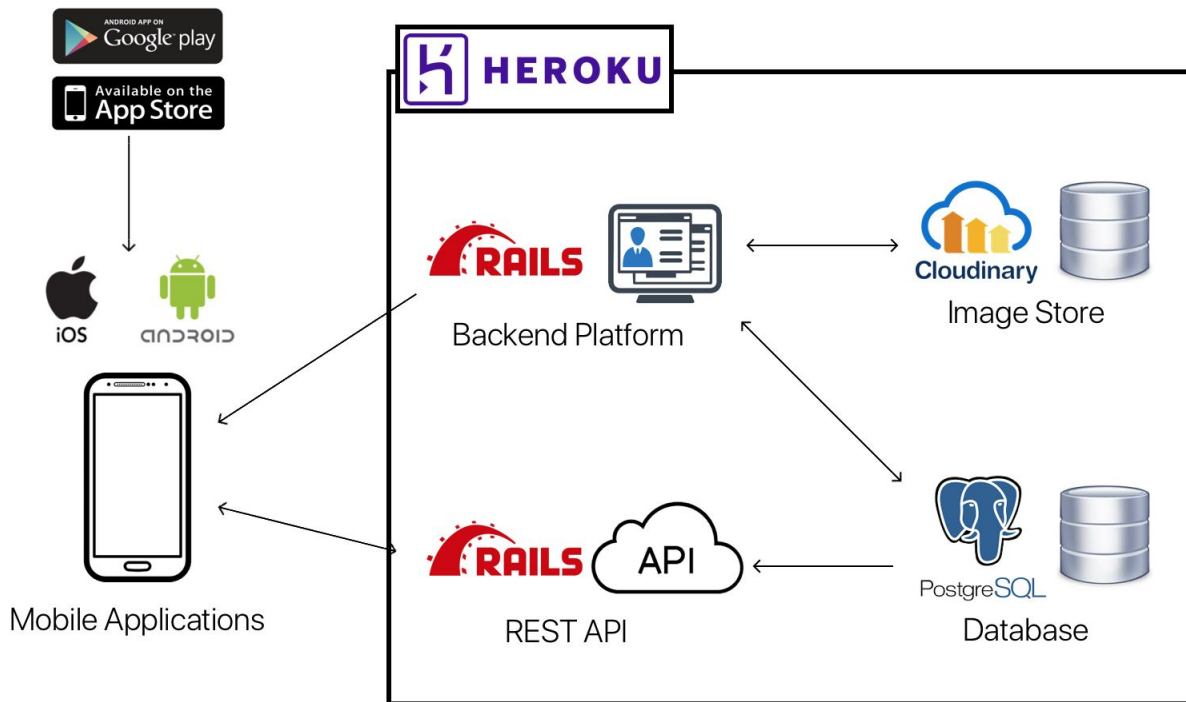
Implementation Language	Swift
Deployment	iOS App Store via XCode, Fastlane build scripts
Libraries Used	CocoaTouch Framework, Turbolinks Framework

2.4.5 - Android Application

Implementation Language	Java
Deployment	Google Play Store, Fastlane build scripts
Libraries Used	Turbolinks Framework

2.4.5 - Technology Overview

Below is a high-level technical diagram of the core components of this project.



2.5 - Resources

2.5.1 - Literature

Ruby on Rails

- Learn Ruby the Hard Way - <http://www.learnrubythehardway.org>
- Ruby on Rails Guides - <http://guides.rubyonrails.org>

iOS

- Apple Developer Portal - <https://developer.apple.com/reference/>

Android

- Android Developer Portal - <https://developer.android.com/>

AWS

- AWS Documentation - <https://aws.amazon.com/documentation/>

2.5.2 - Tooling

- Sublime Text text editor
- Rbenv Ruby environment manager
- XCode IDE for iOS Development
- Android Studio IDE for Android Development

2.5.3 - Hardware

- Development machine - Apple Macbook Pro 15"
- iPhone 6 & iPhone 7 test devices
- Samsung Galaxy S6 test device

2.6 - Definitions, Abbreviations & Acronyms

1.3 Definitions, Acronyms, and Abbreviations

Item	Definition
Racing Driver	An athlete competing in the sport of motor racing.
WebApp	Web Application - a software application accessed via the web browser.
App User	A user of the mobile app, who has registered using their email.
Customer Admin User	The customer, or nominated party who has the ability to edit content on the admin platform.
Super Admin User	The user with the most privileges, this user is responsible for creating new customers.
Cloud	'The Cloud' - a term used to describe a software deployment environment, the hardware of which is not controlled by the system owner. The hardware is abstracted from the system owner, and all code is deployed to the 'cloud'.
iOS	The iOS mobile platform, developed by Apple and deployed on the iPhone range of hardware devices
Android	The Android mobile platform, developed by Google, open source and deployed on many hardware devices manufactured by many companies.
App Store	The iOS App Store - a marketplace where users can purchase and download applications for the iOS platform.
Play Store	The Android Play Store - a marketplace where users can purchase and download applications for the Android platform.
AWS	Amazon AWS - the cloud computing ecosystem provided by Amazon
EC2	Amazon EC2 - servers in the cloud provided within the AWS ecosystem.
Ruby	The ruby programming language, used here in development of the API and admin platform.
Java	The java programming language, used here in the development of Android apps.
Swift	The swift programming language, used in the development of iOS apps.

RoR	Ruby on Rails - The open source web framework used here in development of the API and admin platforms
NginX	Web Server - used in the deployment of the API and admin platforms
JSON	'JavaScript Object Notation' - an object markup language which is the primary means by which the API will return information
XML	eXtensible Markup Language - an object markup language which our API will deliver if requested by the consumer.
WYSIWYG	"What you see is what you get" - used in relation to text editors on websites. A WYSIWYG editor provides an interface similar to Microsoft Word where the user can change the format of text.
Turbolinks	Turbolinks is a technology to make HTML applications faster. It employs body replacement technology to change pages without reloading the page.

3 - System

3.1 - Requirements

3.1.1 - User Requirements

The customer is a racing driver, or their marketing team.

The customer wants a platform with which they can deliver valuable content to the driver's fanbase. This will be done by giving the customer a backend admin platform, where they can add content such as; news stories, social streams, media galleries, and two mobile applications, iOS and Android, which are deployed to the App Store and Play Store respectively.

The customer wants a secure platform, the customer's data will be sandboxed and protected by authentication.

The customer wants to learn about the fanbase, and contact the fans directly. The system will provide a push notification service and email communication to app users, and a view of the registered users with demographic and usage information.

The customer wants a customized app which fits the driver's brand. We will develop a mobile app theming system to allow the customer admin user to modify the app look and feel via the admin platform and deploy the changes in real time to app user's devices.

3.1.2 - Functional Requirements

3.1.2.1 - Overview

3.1.2.1.1 - Admin Platform Functional Requirements

Code	Description
ADM-1	The system should authenticate users and reject unauthorized access.
ADM-2	The system should allow Super Admin users to create new customers
ADM-3	The system should allow Super Admin users to terminate a customer account
ADM-4	The system should allow a Customer Admin user to create new news items
ADM-5	The system should allow a Customer Admin user to edit news items
ADM-6	The system should allow a Customer Admin user to delete news items
ADM-7	The system should allow a Customer Admin user to add social stream details
ADM-8	The system should allow a Customer Admin user to edit social stream details
ADM-9	The system should allow a Customer Admin user to add media items
ADM-10	The system should allow a Customer Admin user to remove media items
ADM-11	The system should allow a Customer Admin user to remove media items
ADM-12	The system should allow a Customer Admin user to broadcast notifications
ADM-13	The system should allow a Customer Admin user to edit the app theme
ADM-14	The system should render a mockup of the mobile app with the theme applied while editing

3.1.2.1.2 - API Functional Requirements

Code	Description
API-1	The system should identify the customer via the Api Key in the headers, and reject all unauthorized consumers
API-2	The system should allow a consumer to fetch customer details which include all the information needed to customize the app look and feel
API-3	The system should allow a consumer to register a new user
API-4	The system should allow a consumer to login a user
API-5	The system should allow a consumer to fetch the list of latest news items
API-6	The system should allow a consumer to fetch the list of gallery items
API-7	The system should allow a consumer to fetch the social feed

3.1.2.1.3 - Mobile Application Functional Requirements

Code	Description
MOB-1	A user should be able to download the application from the platform app store
MOB-2	A new user should be presented with the register page to onboard
MOB-3	The app should store the user details locally to prevent the need for multiple logins
MOB-4	A user should be provided with the interface to login and logout
MOB-5	The app should use the theme provided by the API
MOB-6	The app should have a page showing a list of the latest news items
MOB-7	The app should show a single news item detail view when the user taps on one news item.
MOB-8	The app should have a page showing a list of the latest social feed items
MOB-9	The app should have a page showing the media gallery. The gallery will be separated into an Image section and a Video section. The video section will provide media playback.
MOB-10	The app should provide a view of the live timing data when the driver is participating in a race.
MOB-11	The app should display a message to the user when a push notification is received.

3.1.2.2 - ADM-1 - User Authentication

Description		Priority
The system should authenticate users and reject unauthorized users from accessing the system.		High
Use Case		
Scope	The scope of this requirement is to authenticate every user of the system in order to prevent unauthorized access.	
Primary Actor	A human user.	
Flow Description	A user will be presented with a login dialog, an email field and a password field.	
Precondition	The user is not logged in to the system.	
Activation	The login form is submitted, or a user tries to access any part of the system.	
Main Flow	The user has entered incorrect details into the login dialog. When this happened, the authenticity of the user is checked, and it will fail.	

Alternative Flow 1	The user attempts to access a system component by URL directly. The system will verify the authenticity of the user, which will fail.
Termination	The user is presented with the “401 Unauthorized” page,

3.1.2.3 - ADM-2 - Customer Creation

Description		Priority
The system should allow a Super Admin user to create new customers.		High
Use Case		
Scope	The scope of this requirement is to facilitate the onboarding of new Customers by the Super Admin User.	
Primary Actor	A super admin user.	
Flow Description	The actor will be presented with a form, which contains the various attributes needed to create a new customer.	
Precondition	The actor has a role of “super_admin”	
Activation	The customer create form is validated and submitted.	
Main Flow	The form is submitted and the Customer is created in the database. The customer will be assigned a unique Api-Key which will be used by the apps to identify themselves in the API	
Alternative Flow 1	An error has occurred during the customer creation steps. The actor is presented with a page detailing the error and resumes to the precondition state.	
Exceptional Flow 1	The actor is not a Super Admin user. The actor is presented with a “401 Unauthorized” page.	
Termination	The customer is created and committed successfully to the database.	
Post Condition	The system returns to the super admin home page.	

3.1.2.4 - ADM-3 - Customer Termination

Description		Priority
The system should allow Super Admin users to terminate a customer account		High
Use Case		
Scope	The scope of this requirement is to facilitate the removal of Customers by the Super Admin User.	
Primary Actor	A super admin user.	

Flow Description	The actor chooses the Delete option on the customer detail page. The system will present the actor with an “Are you sure?” message, which they will need to act upon before deletion occurs.
Precondition	The actor has a role of “super_admin”
Activation	The “delete” button is pressed in the Customer detail page
Main Flow	The actor is presented with the “Are you sure?” Dialog, and selects “Yes”. The System will proceed to delete the customer from the database. This will have the effect of disabling the Admin platform for any Customer admin users, and disabling the apps associated with that Customer account.
Alternative Flow 1	The actor is presented with the “Are you sure?” Dialog, and selects “No”. The system returns to the wait state on the Customer detail page. The customer is not deleted.
Exceptional Flow 1	The actor is not a Super Admin user. The actor is presented with a “401 Unauthorized” page.
Termination	The customer is deleted successfully from the database.
Post Condition	The system returns to the Customer detail page.

3.1.2.5 - ADM-4 - News Creation

Description		Priority
The system should allow a Customer Admin user to create new news items		High
Use Case		
Scope	The scope of this requirement is to facilitate the creation of new News items by the Customer Admin user. The interface will include a WYSIWYG editor with which the actor can format the text.	
Primary Actor	A customer admin user.	
Flow Description	The system is in the create news item page. The actor will be presented with fields to enter a title, and a body. The body field will encompass a WYSIWYG editor. The interface will also provide a means to upload a cover image for the story.	
Precondition	The actor has a role of “customer_admin”.	
Activation	The “Add News Item” Button is pressed.	
Main Flow	The actor fills out all necessary information and saves the item. The News item is committed to the database.	
Alternative Flow 1	The actor does not fill out the required information. The actor is presented with a dialog box which will inform of the missing attributes.	

Exceptional Flow 1	The actor is not a Customer Admin user. The actor is presented with a “401 Unauthorized” page.
---------------------------	--

3.1.2.6 - ADM-5 - News Item Editing

Description		Priority
The system should allow a Customer Admin user to edit news items		High
Use Case		
Scope	The scope of this requirement is to facilitate the editing of News items by the Customer Admin user. The interface will include a WYSIWYG editor with which the actor can format the text.	
Primary Actor	A customer admin user.	
Flow Description	The system is in the edit news item page. The actor will be presented with fields to edit the title, and the body. The body field will encompass a WYSIWYG editor. The interface will also provide a means to upload a cover image for the story.	
Precondition	The actor has a role of “customer_admin”.	
Activation	The “Edit News Item” Button is pressed.	
Main Flow	The actor fills out all necessary information and saves the item. The News item is update in the database.	
Alternative Flow 1	The actor does not fill out the required information. The actor is presented with a dialog box which will inform of the missing attributes.	
Exceptional Flow 1	The actor is not a Customer Admin user. The actor is presented with a “401 Unauthorized” page.	
Termination	The news item is committed successfully to the database.	
Post Condition	The system returns to the news list page.	

3.1.2.7 - ADM-6 - Delete News Item

Description		Priority
The system should allow a Customer Admin user to delete news items		High
Use Case		
Scope	The scope of this requirement is to facilitate the deletion of News items by the Customer Admin user.	
Primary Actor	A customer admin user.	

Flow Description	The system is in the news list page. The actor will click the “delete” button associated with the news item. After acting upon an “Are you sure?” dialog, the news item will be deleted or not deleted.
Precondition	The actor has a role of “customer_admin”.
Activation	The “Delete News Item” Button is pressed.
Main Flow	The actor selects “Yes” when prompted. The news item is deleted from the database.
Alternative Flow 1	The actor does not fill out the required information. The actor is presented with a dialog box which will inform of the missing attributes.
Exceptional Flow 1	The actor is not a Customer Admin user. The actor is presented with a “401 Unauthorized” page.
Termination	The news item is committed successfully to the database.
Post Condition	The system returns to the news list page.

3.1.2.8 - ADM-7 - Add Social Stream Details

Description		Priority
The system should allow a Customer Admin user to add social stream details		High
Use Case		
Scope	The scope of this requirement is to facilitate the addition of Social Streams to the platform. The social streams can be Facebook, Twitter or Instagram accounts.	
Primary Actor	A customer admin user.	
Flow Description	The system is in the add social streams page. The user is presented with fields to enter the details for Facebook, Twitter and Instagram social streams. There will be an interface provided to save the information.	
Precondition	The actor has a role of “customer_admin”.	
Activation	The “Save” button is pressed.	
Main Flow	The actor has inserted relevant data in all of the fields. Upon saving, the information will be validated to be correct using the social APIs. After this the details will be committed to the database.	
Alternative Flow 1	The actor fills out bad data in one or more fields, which is identified by the validation step. The actor is presented with some information explaining why the system cannot proceed. The system returns to the pre-save state, with the offending fields highlighted in red.	
Exceptional	The actor is not a Customer Admin user. The actor is presented with a “401	

Flow 1	Unauthorized” page.
Termination	The social details are committed successfully to the database.
Post Condition	The system returns to the home page.

3.1.2.9 - ADM-8 - Edit Social Streams

Description		Priority
The system should allow a Customer Admin user to edit social stream details		High
Use Case		
Scope	The scope of this requirement is to facilitate the editing of Social Streams to the platform. The social streams can be Facebook, Twitter or Instagram accounts.	
Primary Actor	A customer admin user.	
Flow Description	The system is in the list social streams page. The user is presented with an option to edit each item in the list. The user will click the ‘edit’ button, and will be presented with an edit screen to perform changes.	
Precondition	The actor has a role of “customer_admin”.	
Activation	The “Edit” button is pressed.	
Main Flow	The actor has entered the “edit” screen for the chosen Social Stream. The edit screen will have fields to edit the details.	
Alternative Flow 1	The actor navigates to the url directly: ie /social_streams/:ID/edit. This will take the actor directly to the edit page for stream of ID :ID.	
Exceptional Flow 1	The actor is not a Customer Admin user. The actor is presented with a “401 Unauthorized” page.	
Termination	The actor saves the form, and the social details are committed successfully to the database.	
Post Condition	The system returns to the list social streams page.	

3.1.2.10 - ADM-9 - Add Media Items

Description		Priority
The system should allow a Customer Admin user to add media items to the system.		High
Use Case		
Scope	The scope of this requirement is to facilitate the adding of media items to the system. The media items may be images (.png, .jpg etc), or video (.mp4, .mov,	

	.avi etc). The system will take the input file, upload it to a storage service and store the url.
Primary Actor	A customer admin user.
Flow Description	The system is in the media gallery page. The actor selects “Add Item”. The file upload modal will appear and the actor will be prompted to choose the file to upload. The system will then process this file and when done the user will return to the gallery page.
Precondition	The actor has a role of “customer_admin”.
Activation	The “Add Media Item” button is pressed.
Main Flow	The add file modal has appeared. The user chooses a file and the system uploads to to Amazon S3. S3 returns the url for the file, which is saved on the media item model. The system goes back to the media gallery page, with the new item visible.
Alternative Flow 1	The actor closes the modal during the upload phase. In this case the upload will continue in the background, and when the process has finished the media page will update.
Alternative Flow 2	The actor attempts to insert a file which is not supported to the system. The add file dialog will not allow this to happen.
Exceptional Flow 1	The actor is not a Customer Admin user. The actor is presented with a “401 Unauthorized” page.
Termination	The media item information is committed successfully to the database.
Post Condition	The system returns to the media gallery page.

3.1.2.11 - ADM-10 - Remove Media Items

Description		Priority
The system should allow a Customer Admin user to add remove items from the system.		High
Use Case		
Scope	The scope of this requirement is to facilitate the removing of media items to the system.	
Primary Actor	A customer admin user.	
Flow Description	The system is in the media gallery page. The actor selects “Remove Item” on a particular item. The system will throw a dialog to ask the user if they are sure they want to complete this destructive action.	
Precondition	The actor has a role of “customer_admin”.	

Activation	The “Remove Media Item” button is pressed.
Main Flow	The “Are you sure” dialog appears. The user chooses “Yes” on this dialog. The system then removes the media item from the database, and instructs Amazon S3 to destroy the stored media.
Alternative Flow 1	The actor chooses “No” when presented with the “Are you Sure” dialog. The system returns to the media gallery page with no destructive actions performed.
Exceptional Flow 1	The actor is not a Customer Admin user. The actor is presented with a “401 Unauthorized” page.
Termination	The media item information is removed successfully from the database.
Post Condition	The system returns to the media gallery page.

3.1.2.12 - ADM-11 - Broadcast Notifications

Description		Priority
The system should allow a Customer Admin user to broadcast push notifications to app users.		High
Use Case		
Scope	The scope of this requirement is to facilitate the broadcast function using Push Notifications to app users devices.	
Primary Actor	A customer admin user.	
Flow Description	The system is in the broadcast page. The user is presented with a form with fields for Title and Body. The user will fill this form out with the relevant information and submit to send the push notification.	
Precondition	The actor has a role of “customer_admin”.	
Activation	The notification form is filled.	
Main Flow	The actor fills out the push notification form with the title and body to be sent. The actor will then hit the submit button to send a notification to all app users.	
Exceptional Flow 1	An error occurs when sending the push notification. This could be due to an error with the certificates for the push services. The developer will be notified and the user will be informed that the issue will be resolved quickly.	
Termination	The push notification is sent successfully.	
Post Condition	The system returns to the push notification page.	

3.1.2.13 - ADM-12 - Edit App Theme

Description		Priority
The system should allow a Customer Admin user to edit the App Theme.		High
Use Case		
Scope	The scope of this requirement is to facilitate the editing of the mobile application theme via the admin platform.	
Primary Actor	A customer admin user.	
Flow Description	The system is in the edit theme page. The user is presented with a series of fields with colour information for various parts of the app. These fields will include colour pickers for input.	
Precondition	The actor has a role of "customer_admin".	
Activation	The actor changes values in the colour fields on the edit theme page.	
Main Flow	The actor changes the colours required. When the desired theme has been created, the user will click save which will persist the new theme. All app users will experience the new theme on their next launch.	
Exceptional Flow 1	The actor inputs some values which are not valid into a field. The form will display a relevant error message and the user will need to rectify this before saving.	
Termination	The updated theme information is persisted to the database successfully.	
Post Condition	The system returns to the theme edit page.	

3.1.4 - Non Functional Requirements

3.1.4.1 Performance/Response time requirement

The system should be developed with user experience as a priority. Response times should be optimized in all areas to reduce the wait time for the user. In any areas where wait times are necessary, the UI/UX should be developed to help the user understand the wait.

3.1.4.2 Availability requirement

The system should be available as close to 100% of the time as possible. The system will be deployed worldwide, and users may need to access the system around the clock.

3.1.4.3 Recover requirement

In the event of a failure, the system should have the ability to recover and restart itself.

3.1.4.4 Security requirement

The system should utilise the most modern security features available to protect the users data, and also the deployment data. Sensitive user information should be encrypted. Unauthorized

users should be prevented from entering any parts of the system which could uncover sensitive data.

3.1.4.5 Reliability requirement

The system should be reliable and bug-free. Under normal circumstances system crashes and unexpected behaviour should not be tolerated.

3.1.4.6 Maintainability requirement

The system should be developed using the software best practices laid out by the creators and community of RoR. The code should be well commented in order to aid any future developer who may be maintaining the system.

3.1.4.7 Portability requirement

The system should be developed in such a manner that it can be deployed in a variety of environments. Over time the correct environment may change, due to reasons such as cost, scaling,

3.1.4.8 Extendibility requirement

The system should be built in such a way that future features can be added to the system in the most efficient manner. The system should be modular in nature, and individual components can be swapped and/or upgraded.

3.1.4.9 Reusability requirement

The system should be built using the DRY principle. Where possible components should be reused throughout the system.

3.1.4.10 Database Requirement

The database should be scalable. From the beginning thought should be put into all queries with respect to their execution on much larger datasets. The future possibility of segmented data should also be entertained.

3.1.5 - Graphical User Interface Requirements

3.1.5.1 - Interface Elements

The system will utilise many different forms of input interface elements. The aim is to provide the user with the best tools for the job and make working within the system as easy as possible.

3.1.5.2 - WYSIWYG Editor

In the news section, the user will be provided with a WYSIWYG editor for input. This editor will feel familiar to the user if they are familiar with common text processing tools. The editor will include modes to enter normal text with formatting tools, enter text in markdown syntax, or enter raw html.

3.1.5.3 - Color Picker

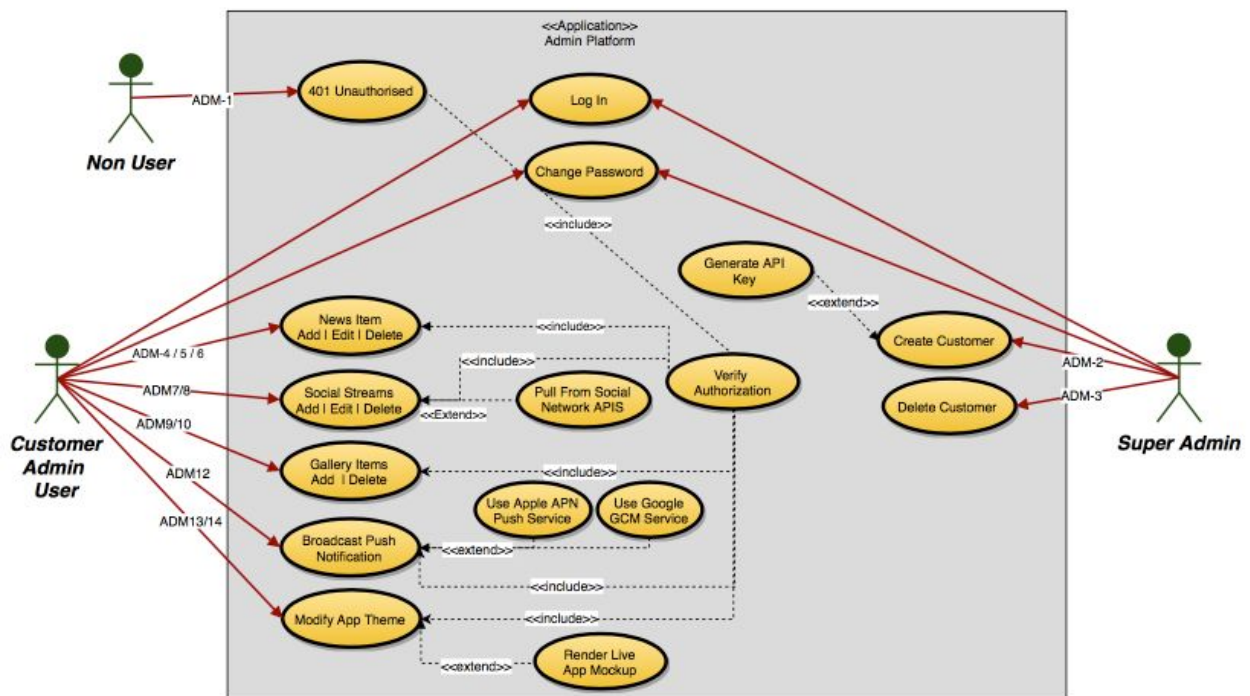
In the theme editor, the user will have two options for entering the color values. They may enter the value directly into the form as a HEX code (#RRGGBB), or they may use the color picker. The color picker will present a modal color picker which will allow the user to visually choose the color.

3.1.5.4 - Modal Views

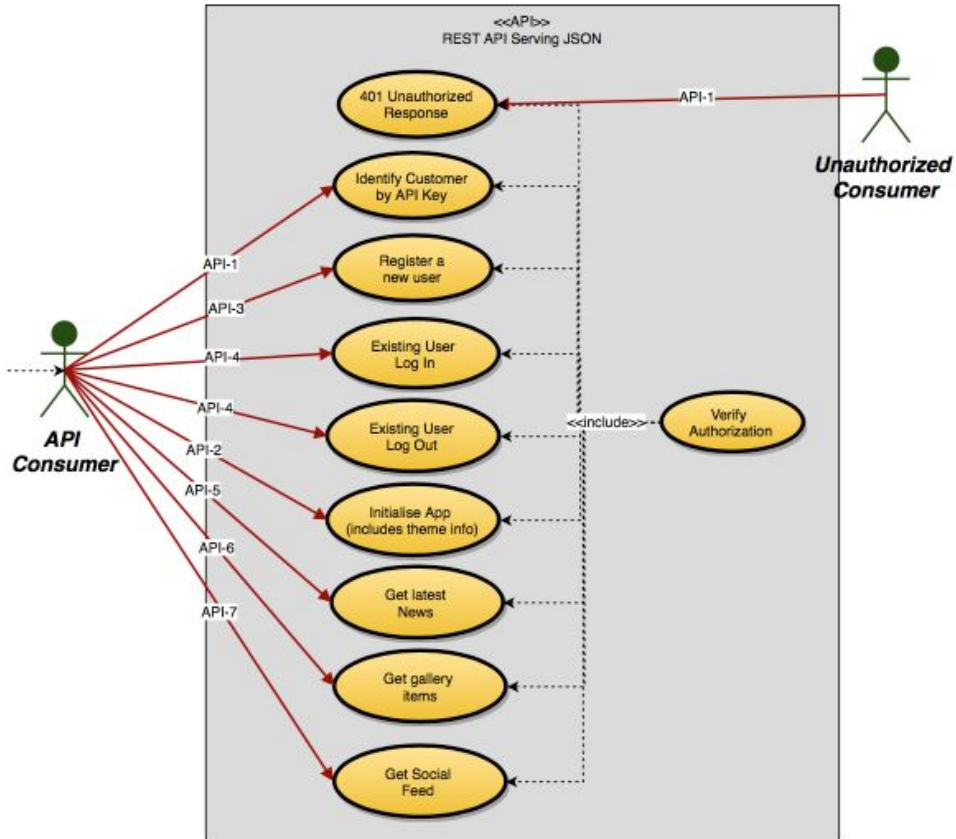
Where possible, and where sensible, modal views should be used to present information and input forms. This allows us to cut down on user wait times as page reloads are not needed.

3.2 - Use Case Diagrams

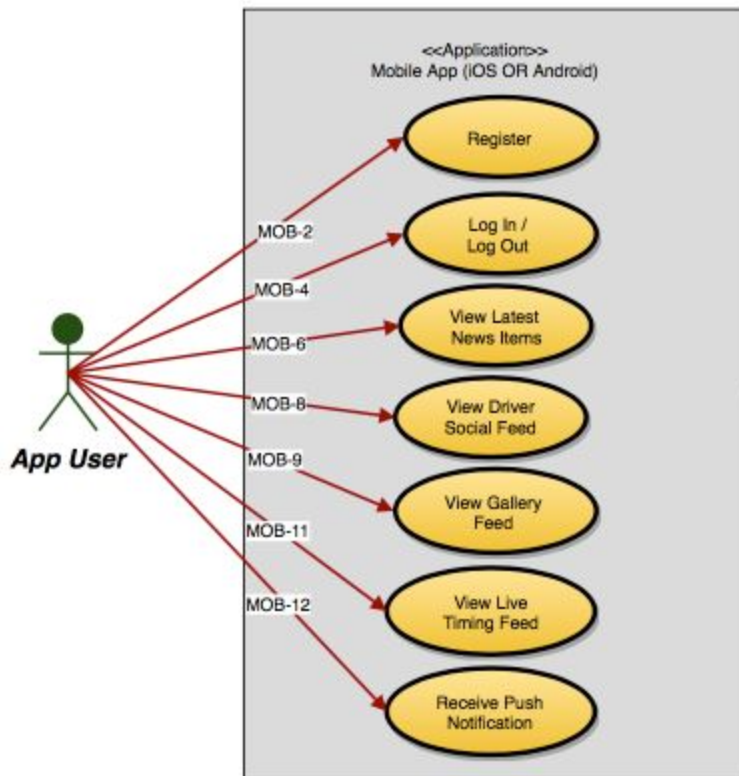
3.2.1 - Admin Platform Use Case Diagram



3.2.2 - API Use Case Diagram



3.2.3 - Mobile Application Use Case Diagram



3.3 - Implementation

3.3.1 - Backend Platform

The backend platform is implemented using Ruby on Rails (RoR). The application is a RoR application utilising a Postgresql database. The Postgresql database was chosen due to the deployment method - Heroku. Heroku requires that Postgresql is used. For this reason I also utilised Postgresql locally in order to ensure that my local and production environments were as similar as possible.

The platform utilises the Model View Controller (MVC) pattern. MVC ensures separation of concerns and allows me to adhere to the DRY (Don't Repeat Yourself) principle. The model layer defines the database schema and facilitates communication with the database via ActiveRecord. The Controller layer sits in the middle of the Model layer and the View layer. The Controller layer is responsible for all communication with the Model layer, and rendering of the views. The controller layer passes any data required to the view. The view cannot communicate directly with the Model layer. This keeps the views lightweight, and speeds up rendering times.

I utilised the haml templating language to produce the views in the application. The haml language allows me to write HTML with ruby language embedded. This way I can surface ruby variables throughout the code to provide dynamic views. Haml is very fast to write, as there are no closing tags. This allowed me to iterate quickly on my views and produce designs faster.

3.3.2 - Mobile Applications

The mobile applications are hybrid applications utilising the cutting-edge Turbolinks 5 technology. The app utilises a fully native navigation, and renders turbolinks-enabled views which are rendered by the API. This results in what feels to the user as a native application, but the flexibility of a web application. As the views are rendered from the backend, changes can be made quickly. Another upside of this is that the main content is cross-platform.

The applications themselves, being native, are developed in the platform's native languages. The iOS application is developed using Swift in the XCode IDE, while the Android application is developed using Java in the Android Studio IDE.

The main content of the app is HTML5/CSS and Javascript delivered via Turbolinks, but the theme engine is implemented using JSON. When the app launches, it fetches the theme colours from the API. When the colours are fetched, they are stored in local storage in order to speed up future app launches. The app will query the API at strategic times in order to fetch the latest theme colours and render the app appropriately.

3.3.3 - REST API

The REST API is a separate part of the main Rails application. The API delivers content in both HTML and JSON. The endpoints in use are:

Endpoint	Description	Method	Reponse
/api/v1/post	News items index	GET	HTML
/api/v1/post/:id	News item detail	GET	HTML
/api/v1/bio	Get Bio	GET	HTML
/api/v1/calendar	Calendar index	GET	HTML
/api/v1/media	Media index	GET	HTML
/api/v1/media/:id	Media item detail	GET	HTML
/api/v1/theme	Get theme	GET	JSON

3.3.4 - Theme Engine

The theme engine is the heart and soul of the mobile applications. In order to make white label apps work for a large number of customers, it is necessary to provide customisation. By allowing the user to choose the colour scheme for their app, I can provide a great deal of customisation. The user can select the colours for their app on the backend platform using the colour pickers, and the results are instantly deployed to all mobile users instantly.

3.3.5 - Broadcast

With a large user base comes great potential for getting your story out. A big addition to this is the inclusion of push notifications to the users. The customer can send out a push notification to all users of the applications from the backend platform. Push notifications can also be automatically configured to be sent out when a news story goes live.

3.3.6 - White Labeled Apps

When developing the functionality for the mobile applications, I developed all functionality inside a library. This library is reusable, and therefore when I need to onboard a new user, the creation of the mobile apps is streamlined. I need to simply include my RFBASE library and add the Customer API Key to the project. The result is a ready to go white labeled native mobile application.

3.3.7 - Deployment

I chose heroku as my platform for deployment. I chose Heroku for a number of reasons:

- Heroku is cost effective - this platform will cost ~\$7.00 per month to keep running at the current scale.
- Heroku is simple to use - deployments can be carried out with a simple command.
- Heroku has many plugins - such as Cloudinary for asset storage, and PostgreSQL for data storage.

I found the deployment to very simple, and encountered no major roadblocks. If I notice any bugs I can make a change in the code and deploy instantly to production with just one command. To deploy the application, run the following command:

```
git push heroku master
```

This pushes your latest changes to the Heroku branch. When the heroku branch is updated, a new build happens automatically, and your app is deployed on successful build. If there is any issue with the build the deploy will not happen, and your current production environment remains unchanged.

3.3.8 - Security

As the platform employs multi tenancy, security is a high priority. All data is keyed by customer ID, and this data can only be surfaced to a user of that customer. The API requires that a Customer API-Key is sent in every request in order to receive data. This key is a hash generated when the customer is created.

For security, I followed the guidelines outlined in the OWASP Top Ten [2] to ensure I left no security holes while developing the application.

3.4 - Functionality

3.4.1 - News

The platform user can create news posts to deliver stories to the app users. The news posts consist of a title and a body, a go-live date and two images - the cover photo for wide view on top of the post detail page, and a square image for use on the item list in the index page. The News item index page on the platform shows the title, body, live date and controls to edit, delete and show more for each item. The item detail page shows the cover photo along with the story itself.

The news stories are shown to the mobile application with the newest posts to the top of the list. Posts do not appear on the mobile apps until the live date has been reached.

3.4.2 - Bio

The bio contains many info points about the driver. The fans want to know as much as possible about their favourite driver, and the bio is the place to deliver that information. The following information is configurable and is shown to the app users:

- Profile Photo
- Name
- Date of Birth
- Hometown
- Facebook
- Twitter
- Instagram
- Website

- Career

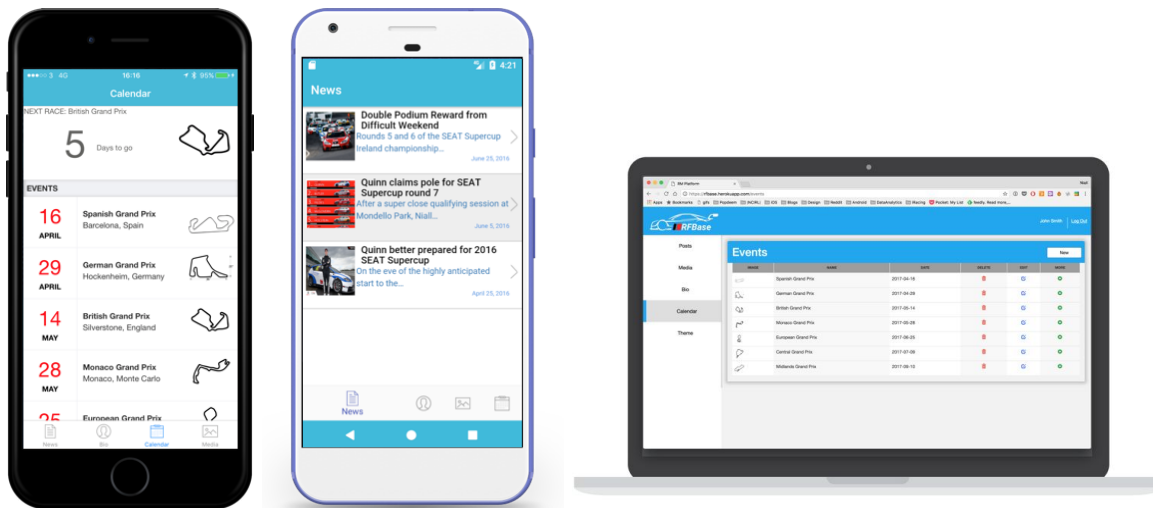
The career section is a WYSIWYG editor which allows the user to create a formatted prose to explain the entire race career of the driver.

3.4.3 - Calendar

The calendar allows the platform user to input the race season details. An event consists of a track image, event name, description and the date it will happen. On the platform, the user has functionality to add, edit and delete events. These events are shown to the app users in chronological order. There is a “Next Race” module at the top of the page, which shows a daily countdown to the next race.

3.4.4 - Theme

The Theme allows the customer to choose four custom colours, which are used throughout the mobile applications. Using the theme engine the customer can choose a unique colour scheme to show their own brand or nationality. The engine uses color pickers on the front end and a live app render, to show how the scheme will look. Upon saving the scheme. The mobile applications are updated instantly.



4. Testing

4.1 - Usability Testing

4.1.1 - Trunk Test

The trunk test is a method of determining if key elements are present in a products navigation. There are five main headings under which to perform the trunk test:

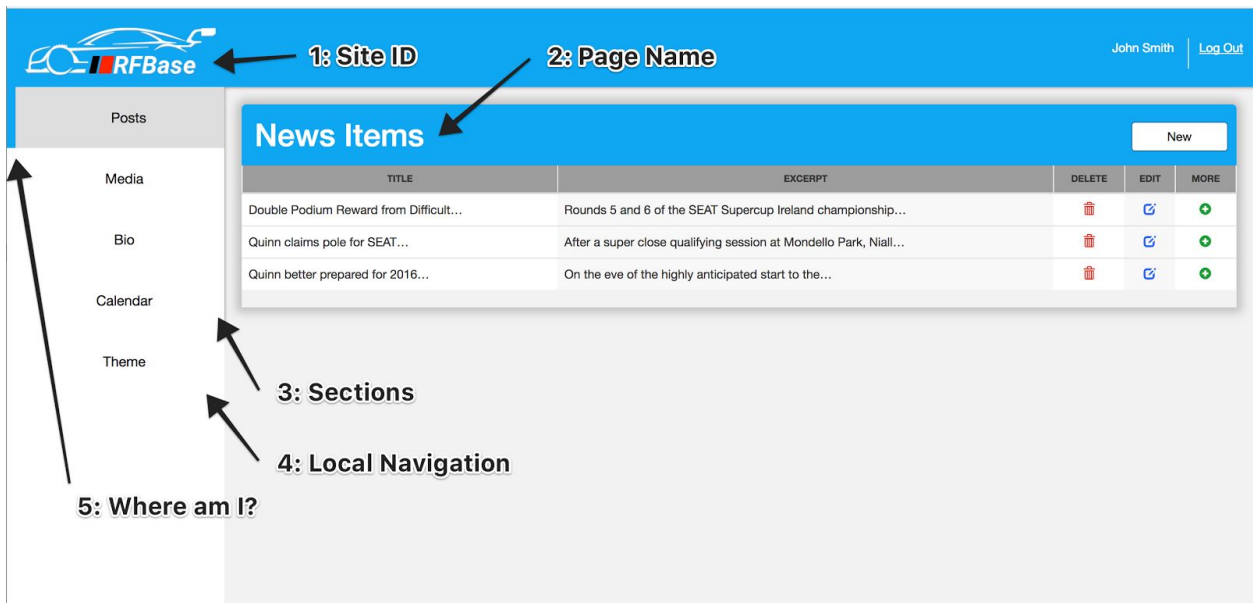
1. Site ID
 - a. Are there visual cues to let the user know that they are on RFBASE?
2. Page Name
 - a. Are there visual cues to let the user know which page they are on?
3. Sections

- a. Is it clear to the user what the main sections are on the product.
- 4. Local Navigation
- 5. Where am I
 - a. Is there a “You are here” indicator?

Trunk Test Results:

Metric	Web Platform	Mobile App
Site ID	There is an “RFBBase” logo on the top left of the navigation bar on each page of the website. This coupled with the user name on the top right will let the user know they are on the RFBBase platform, and signed into a particular Customer profile.	The mobile app is branded with the driver's personal app icon. When the app launches, there is a branded splash screen also. It is not common practice to include a logo once inside a mobile application, so I have followed these guidelines with this product.
Page Name	Each section in the web platform has a large H1 on the top of the content area which indicates to the user which section they are in.	When navigating through the mobile apps, the title bar always shows the name of the current section on the top of the screen.
Sections	The left hand navigation surfaces all section names to the user.	The bottom tab-bar surfaces all section names to the user.
Local Navigation	The web platform employs a left-hand navigation bar which allows the user to navigate throughout the website.	The mobile apps employ a bottom tab-bar navigation system which allows the user to navigate throughout the application.
Where am I?	There is a visual indicator on the left side navigation which shows the user which section is currently shown. This along with the page name leaves no room for ambiguity.	The selected section is highlighted in the tab bar with the theme primary colour. This, along with the page name leaves no room for ambiguity.

Web Platform:

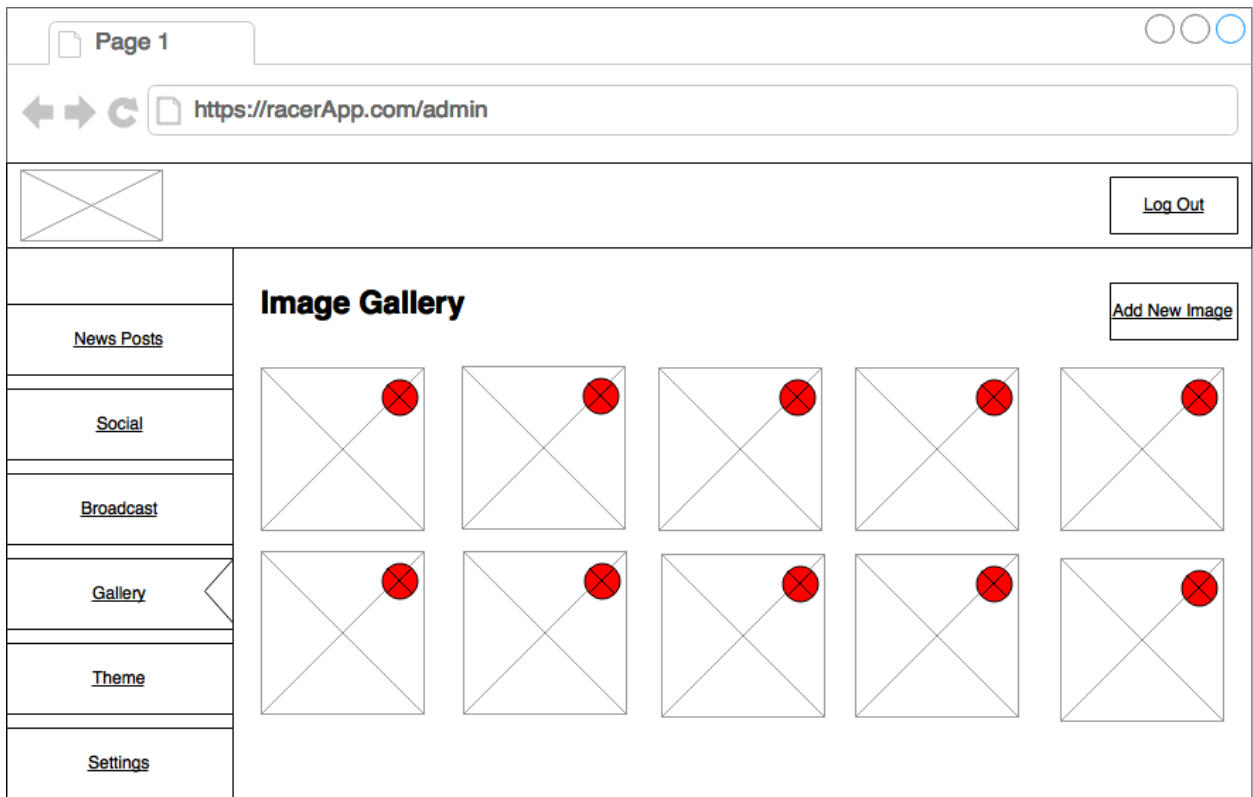


Mobile Applications



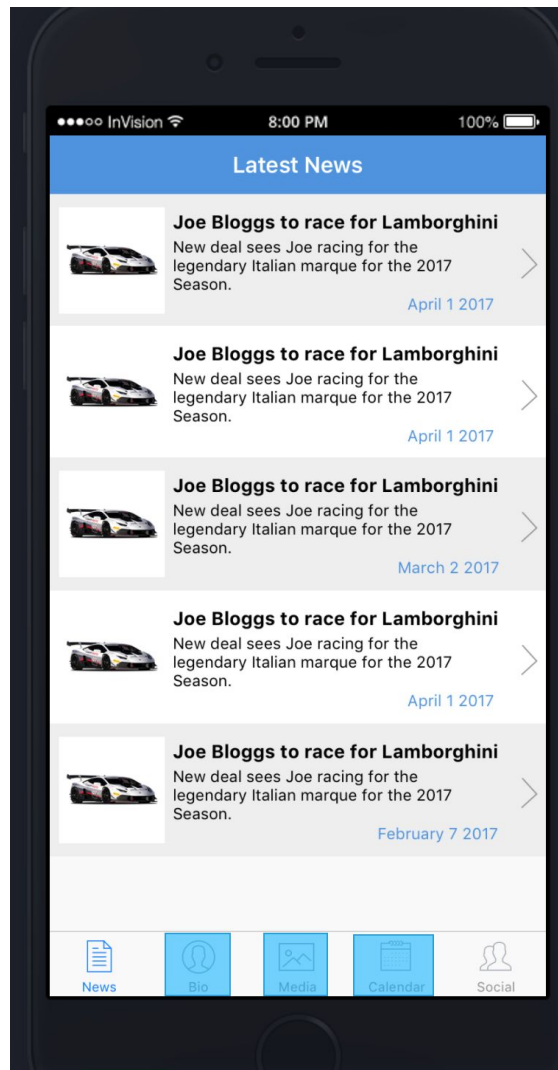
4.1.2 - 5 Second Test

For this test, I showed five testers an image of the web platform, and after five seconds I removed the image. I then asked them a series of questions about the website. The purpose of this test is to determine if the site purpose is instantly clear. At the highest level, the results of this test were, as expected, not very positive. This is due to the fact that this is a very specific service. It is safe to assume that any user will know what the website is designed to accomplish. I believe that a user would need to be informed of the value proposition before performing the 5 second test.



High Fidelity Prototype

Prototype can be viewed here: https://invis.io/JSBM6JNDP#/232662379_Latest_News



4.2 - Unit Testing

I used unit testing in the Ruby on Rails application to ensure all functionalities are working as expected. When run, the test suite runs tests on the model and controller layers to test the functionality of the following:

1. News item - Create, Update, Delete
2. Media Item - Create, Update, Delete
3. Bio - Update
4. Calendar Item - Create, Update, Delete
5. Theme - Modify and Save
6. User - Create, Update, Delete, Reset Password

Running the Tests

Locally:

Navigate to the project root and run the following command:

```
bundle exec rake test
```

Production:

Navigate to the project root and run the following command:

```
heroku run rake test
```

4.3 - Integration Testing

The backend platform and the API are covered by the Ruby on Rails unit tests. The whole system as a unit is then tested using the mobile applications. I enlisted three testers to test my system. These testers came from different backgrounds: one is a technical product manager for a large company, another an engineer for a large race team, and the other a tech enthusiast. This allowed me to test the product from a technical point of view, and from the perspective of a user in the motorsport community.

For this test, I supplied the testers with a short introduction to the product and their own personal login details. I supplied them with a version of the mobile application. I informed them that they could use the backend platform to modify the content and view these changes in the mobile application. They could also use the backend application to modify the theme colours and test the theme engine.

I sent questions to the testers in the form of a Google Form, and captured the responses in a spreadsheet.

Testers

Name	Paul Hays
Company	Independent Media
Position	Product Manager

Name	Graham Quinn
Company	Andretti Autosport
Position	Race Engineer

Name	Adam Smith
Company	Future Finance
Position	Manager

Name	Kevin O'Hara
Company	LOH Motorsport

Position	Race Driver
-----------------	-------------

Results

The form can be viewed at the following link: <https://goo.gl/forms/PXLix3JWMUZOLm92>

Question	Paul Hays	Graham Quinn	Adam Smith	Kevin O'Hara
Platform				
Was it immediately obvious what type of content was presented to you on landing?	Yes	Yes	Yes	Yes
The site navigation is effective (1-5 scale)	5	5	5	5
It is clear which Items on the platform on which I can perform a DELETE operation	5	4	5	4
It is clear which Items on the platform on which I can perform an EDIT operation	5	4	5	4
It is clear which Items on the platform on which I can view detail	5	4	4	4
The WYSIWYG (What you see is what you get) editor is a good addition for styling text in News and Bio	5	5	5	5
I find the site aesthetically pleasing	Yes	Yes	Yes	Yes
Feedback - Please supply any feedback	I really like the use of ux icons for the delete button with obvious colouring. Everything is very clear and structured to use. Difficult to get lost and not perform the desired task or action. Performs as expected. Speed of use is good too. Image uploaded very	Nice simple design and layout. Easy to find my way around.	Easy to navigate and content is easy to input.	I really like the theme engine. I had fun playing around with my personal colours and seeing them update on the app.

	quickly. WYSIWYG works well and is definitely the right choice for this format. Hard to find fault as app performs as desired to match the user needs.			
Mobile Application				
It is clear which sections are available and how to navigate to each section	5	5	5	5
It is clear how to show the detail for a news item	5	5	5	5
The social links on the bio page are a good addition	4	4	4	4
The layout of the media page is user friendly.	4	4	4	4
I find the app aesthetically pleasing	5	5	5	5
The theming engine is a good addition to the application	5	5	5	5
Feedback - Please supply any feedback	Theming engine is great to have so can individual colours to match racing theme or nationality.			

Conclusion

The results from these tests were largely satisfying. I am happy that my choice of UX was easy for the testers to use and there were no issues. There were not many red flags in the results, perhaps the layout of the Media page on mobile can be tweaked to provide a better experience. I am very happy that the theme engine got such great reviews.

4.4.1.2 - Social Page

The screenshot shows a web browser window with the address bar containing `https://racerApp.com/admin`. The page has a header with a placeholder for a logo and a [Log Out](#) button. A sidebar on the left contains menu items: [News Posts](#), [Social](#) (which is active), [Broadcast](#), [Gallery](#), [Theme](#), and [Settings](#). The main content area is titled "Social" and contains three input fields for "Twitter Handle", "Instagram Handle", and "Facebook Page". A [Save](#) button is located below these fields.

4.4.1.3 - Broadcast Mockup

The screenshot shows a web browser window with the address bar containing `https://racerApp.com/admin`. The page has a header with a placeholder for a logo and a [Log Out](#) button. A sidebar on the left contains menu items: [News Posts](#), [Social](#), [Broadcast](#) (which is active), [Gallery](#), [Theme](#), and [Settings](#). The main content area is titled "Send a push notification to your users!" and contains two input fields: "Title" and "Body". A [Send](#) button is located below the "Body" field.

4.4.1.4 - Gallery Mockup

Page 1

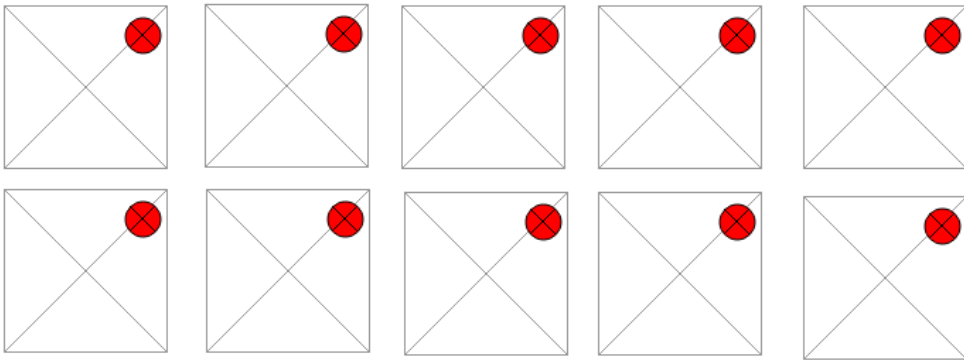
https://racerApp.com/admin

Log Out

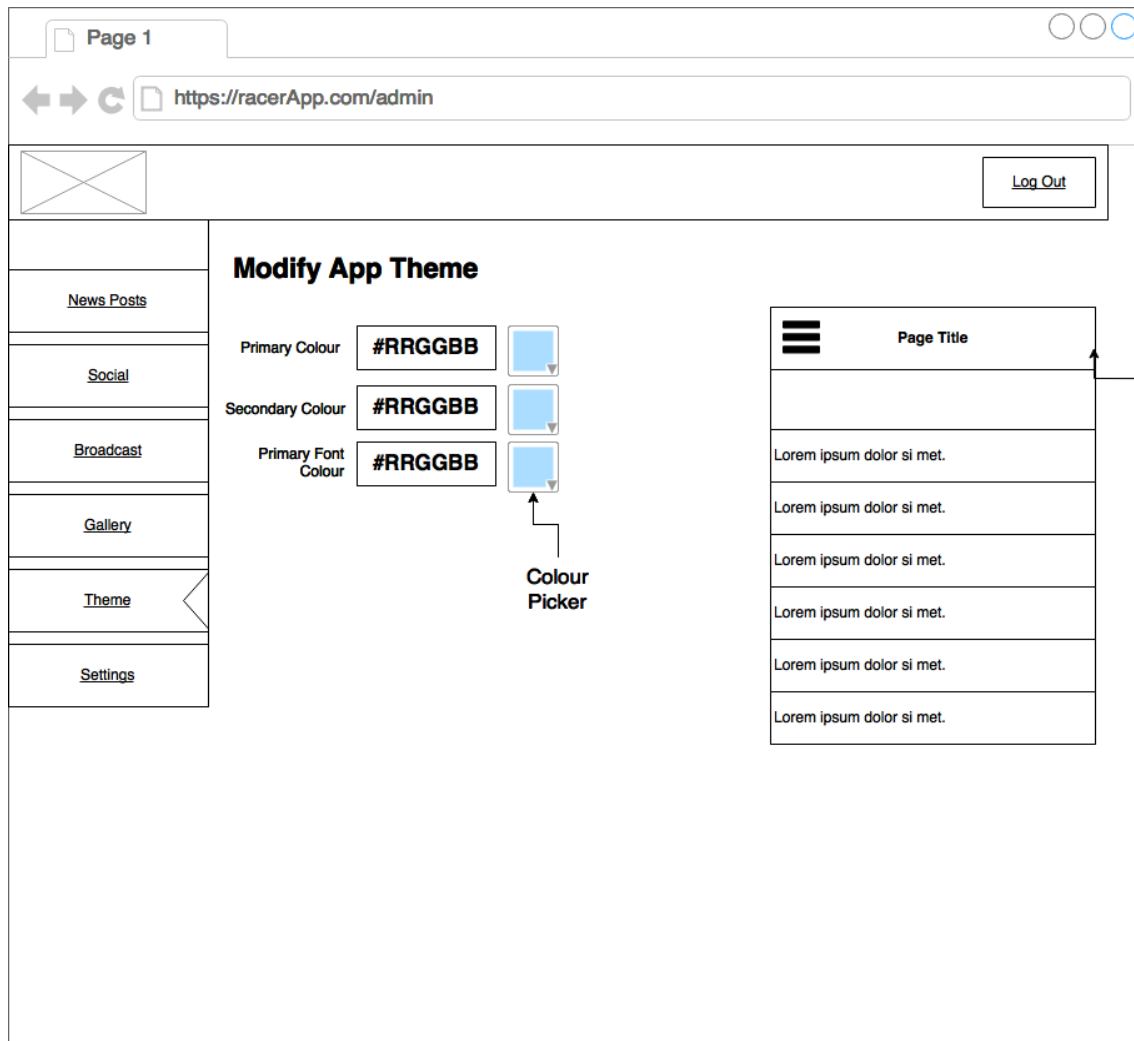
Image Gallery

Add New Image

- News Posts
- Social
- Broadcast
- Gallery**
- Theme
- Settings



4.4.1.6 - Theme Page Mockup



4.5 - Application Programmers Interface (API)

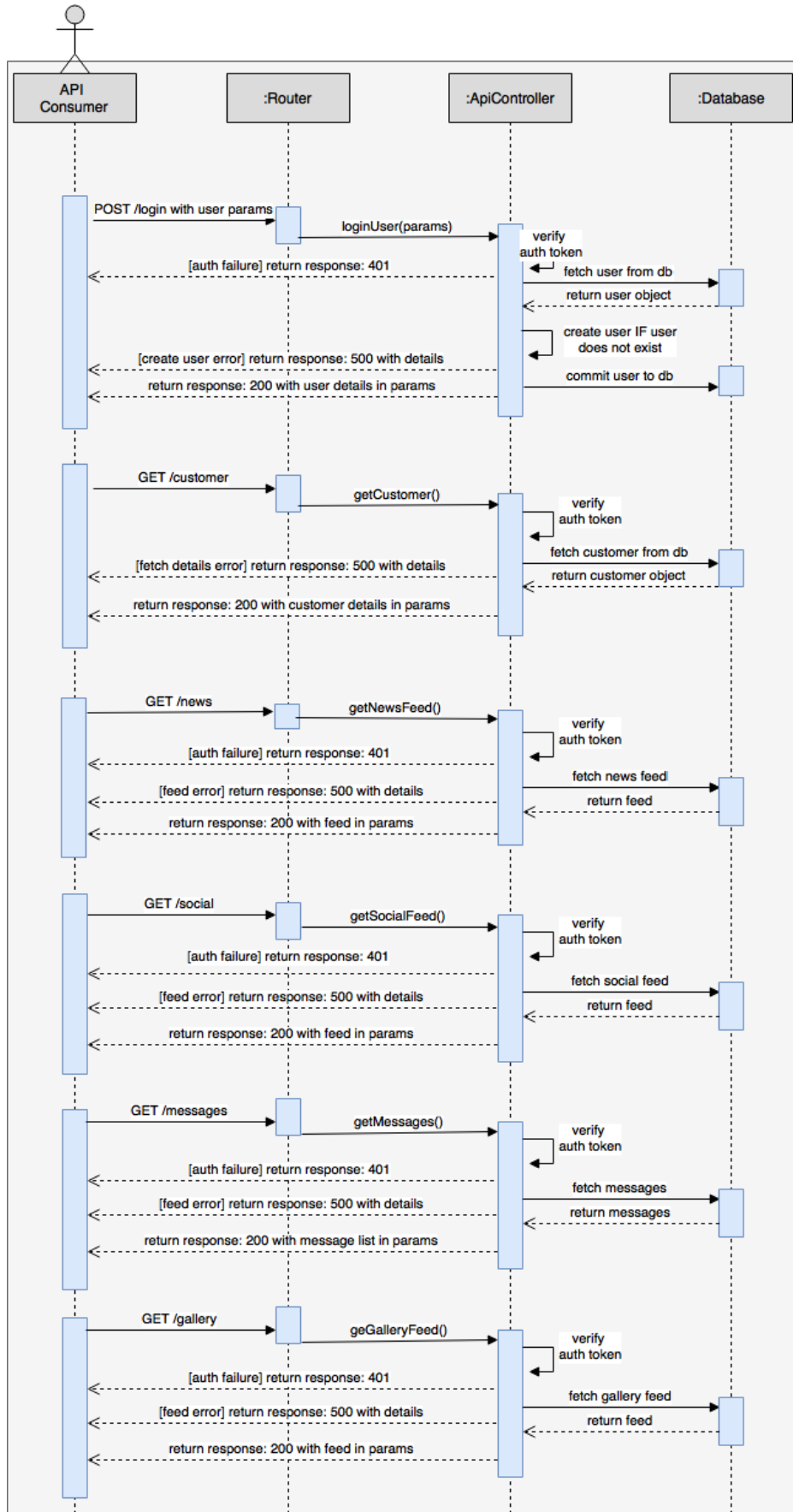
4.5.1 - API Specification

The system will include a REST API for communication with the mobile applications. The API will consume and produce JSON, and support the standard HTTP methods; GET, POST, PUT, DELETE. The API will be robust and provide detailed responses when the desired action is not completed. The endpoints to be created are listed below:

Endpoint [METHOD : endpoint]	Description
POST : /login	Perform the login action
POST : /signup	Perform the signup action
GET : /customer	Retrieve customer details
GET : /news	Retrieve list of news items

GET : /social	Retrieve list of social items
GET : /gallery	Retrieve list of gallery items
GET : /messages	Retrieve list of messages

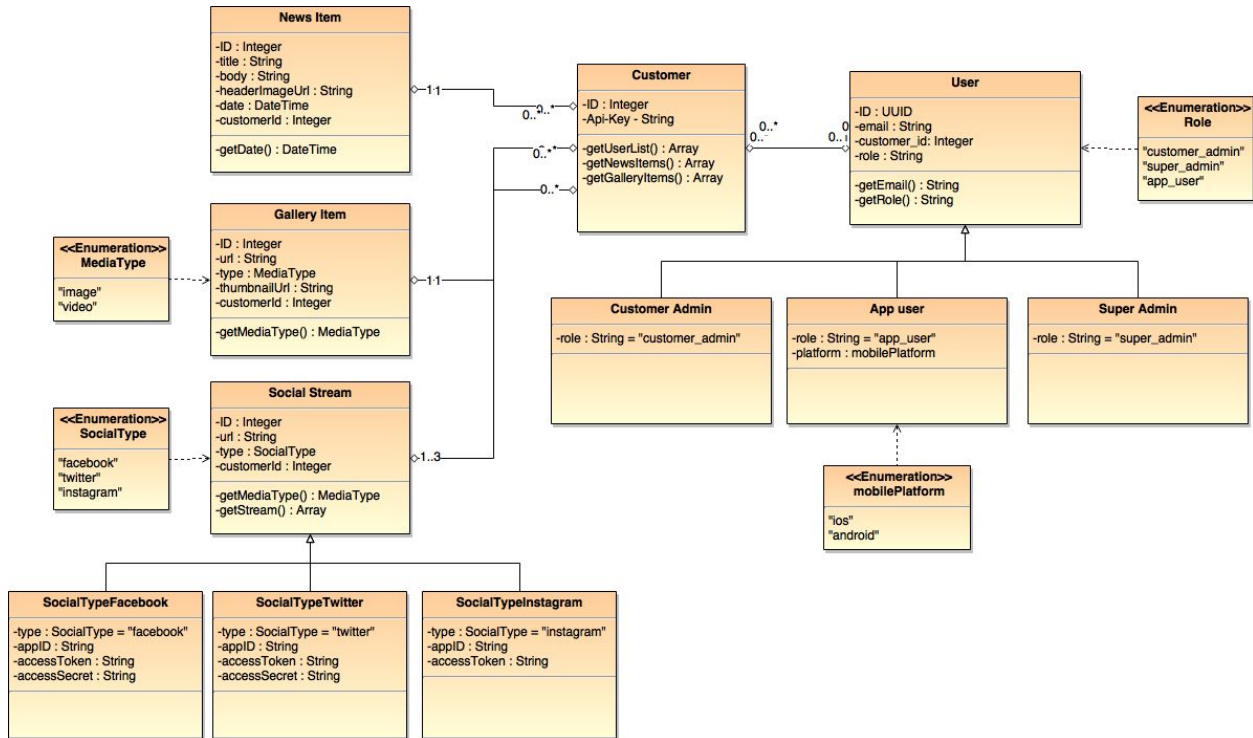
4.5.2 - API Sequence Diagram



4.6 - Architecture

The core system will be built in Ruby on Rails. The system should be built to the best practices of the RoR platform. The database used will be MySQL. Care should be taken to ensure the system is scalable in the long term.

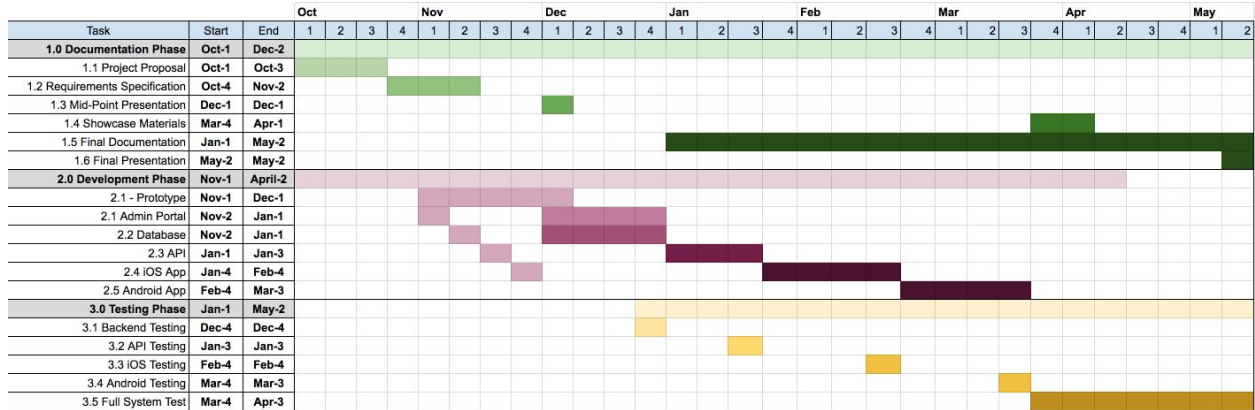
4.6.1 System Class Diagram



5 - Project Plan

The project was split into three main phases: the documentation phase, the development phase, and the testing phase. According to the deliverable dates set out by the NCI faculty, there will be some overlap in these phases. See item 5.1 - *gant chart* for the exact spread throughout the project.

5.1 Gantt Chart



6 - System Evolution

The system will launch with all of the requirements set out in this document in place. There is scope for further evolution in some key areas. As we gather user feedback over time, new media types and outlet types may be added to the system. The mobile applications should constantly evolve, as the mobile platform ecosystem is currently in a period of rapid improvement. Every yearly cycle brings new APIs and capabilities. We should always be exploring these areas to bring new features to the end user.

As time goes on and I begin to see user patterns in the application, areas to improve will show up. I see some distinct areas for new functionality going forward:

- The ability for an app user to follow/like the drivers social accounts from inside the application.
- Live timing streamed to the application during races.
- A results section.
- Video support in media.
- Expand beyond race drivers to other athletes.

With more time and resources I can see this project becoming a viable business. There is a market of racing drivers and other athletes who want to interact directly with their fans.

7 - Useful Links

Platform Github Repository	https://github.com/NQuinn27/RFBBase
iOS Github Repository	https://github.com/NQuinn27/RFBBase-iOS
Android Github Repository	https://github.com/NQuinn27/RFBBase-Android
Platform	http://rfbase.herokuapp.com

Login Details for Platform
Email: niall+johnSmith@niallquinn.me Password: johnSmith

8 - References

1. "Turbolinks/Turbolinks". *GitHub*. N.p., 2017. Web. 9 May 2017.
2. "Category:OWASP Top Ten Project - OWASP". *Owasp.org*. N.p., 2017. Web. 9 May 2017.

Appendix 1 - Project Proposal

Project Proposal

RFBBase

Niall Quinn

X13108727

niall.quinn1@student.ncirl.ie

BSc (Hons) in Computing
Evening

Software Development Stream

Table of Contents

- 1. Executive Summary
- 2. Background
- 3. Technical Approach
 - 3.1 Technical Overview
- 4. Resources
 - 4.1 Literature
 - 4.2 Tooling
 - 4.3 Hardware
- 5. Project Plan
 - 5.1 Gantt Chart
- 6. Technical Details
 - 6.1 - Backend Platform
 - 6.2 - API
 - 6.3 - Database
 - 6.4 - iOS Application
 - 6.5 - Android Application
- 7. Evaluation
 - 7.1 - Unit Testing
 - 7.2 - Manual Testing
 - 7.3 - Requirements Evaluation

1. Executive Summary

1. To produce a highly functional marketing platform for racing drivers.
2. To supply valuable functionality to target and grow an audience.
3. To provide this using modern languages, frameworks and tooling.
4. To develop a custom theme engine to produce configurable look-and-feel for the mobile applications.
5. To leverage third party services to provide valuable real-time data to end users.

2. Background

Motor racing drivers today spend a large amount of time marketing their brand to the large motor racing fan base. At the moment, drivers use traditional social media, Twitter, Facebook, Instagram etc as a means to reach their audience. There is a place in the market for a single outlet from which the driver can target and grow their audience.

A customer on the platform will receive a login to their own backend portal, where they will be able to add various information such as: News, Race Calendar, Images & Video and Social streams. All of this information will be served to their own custom-themed iOS and Android apps which will be deployed via the iOS App Store and Google Play Store respectively.

The business model for this product is a SAAS model. A Customer on the platform will pay an upfront development fee, and then pay a monthly subscription to use the service.

I see a gap in the market for this product, and am confident that when deployed will be a viable product.

3. Technical Approach

This project will be developed using modern languages, frameworks, tooling and infrastructure. A clear and concise project plan has been set out in order to maximise time available and ensure all parts are developed in time, and integrate seamlessly. At regular intervals, progress will be assessed and the project timeline updated accordingly.

From a high level, this project will be comprised of five main components:

1. Backend portal - a Ruby on Rails Application
2. API - a Ruby on Rails API
3. Database - a MySQL database using Amazon RDS
4. iOS Application - native application developed using Swift
5. Android Application - native application developed using Java

I will endeavour to identify all requirements in the requirements specification stage, in order to build the platform from the beginning with all entities and relationships in scope. This will lead to less headaches in the testing and integration phases later.

The project will employ a continuous integration scheme, with automated unit testing. All code merges to the master branch will pass through the automated testing framework, with bad builds being rejected. This will ensure that critical bugs do not make it to the production code.

The project will make use of the Elastic Beanstalk platform on AWS. This allows us to choose the most cost-effective configuration to begin with on AWS. If the load on the app or API becomes too large, Elastic Beanstalk will automatically create new instances of the application to handle this load. This provides us with an extra layer of robustness to scaling problems.

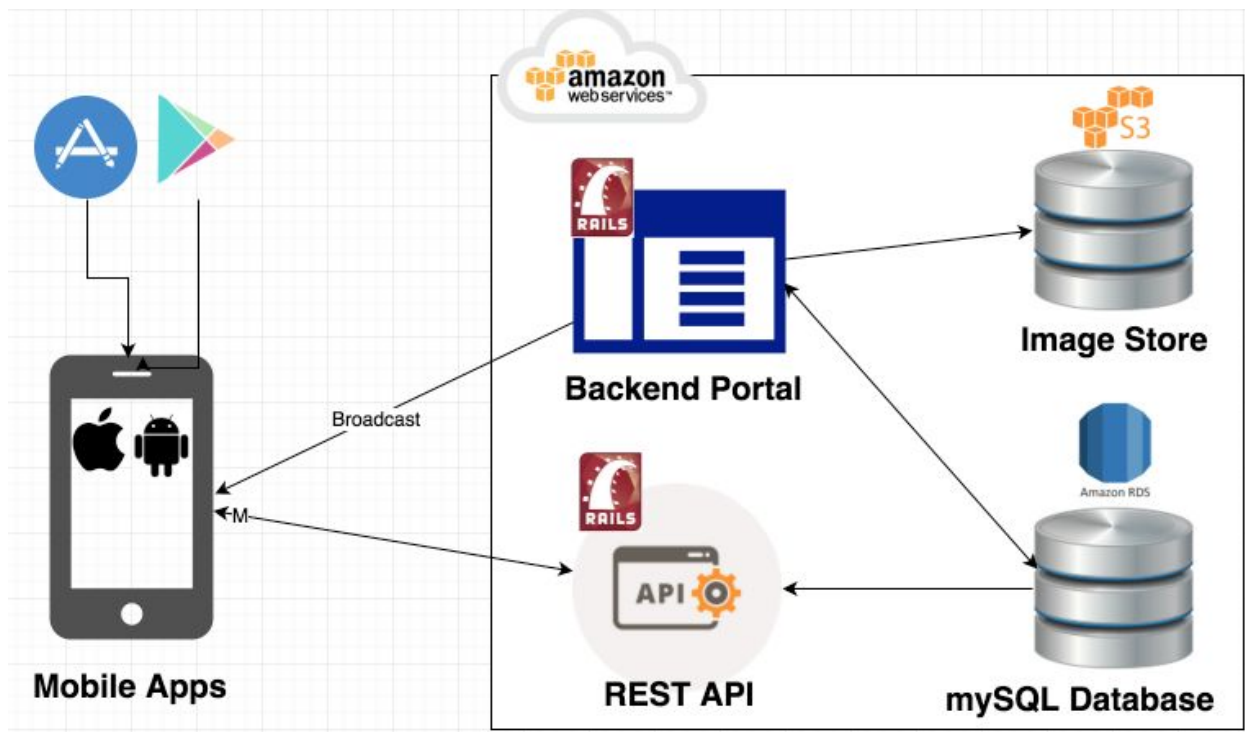
A key part of this product will be the custom theming engine for mobile. The mobile applications will pull down a specially formatted JSON file from the API. Acting upon this file, the application will initialize core colours, fonts and images. This results in each of the platform customers receiving a customized application, to suit their own brand.

The latest technologies in automated building will be used on the mobile apps, to reduce developer input when onboarding new customers. The *Fastlane* platform will be used to execute custom scripts which will take care of app building and distribution. This will be a key area to focus on as the business scales in order for customer onboarding to not become a headache.

Requirements will be captured by interviewing current racing drivers. I will ask them what they would like to see in the application and build up a ranking of features from those interviews.

3.1 Technical Overview

Below is a high-level technical diagram of the core components of this project.



4. Resources

4.1 Literature

Ruby on Rails

- Learn Ruby the Hard Way - <http://www.learnrubythehardway.org>
- Ruby on Rails Guides - <http://guides.rubyonrails.org>

iOS

- Apple Developer Portal - <https://developer.apple.com/reference/>

Android

- Android Developer Portal - <https://developer.android.com/>

AWS

- AWS Documentation - <https://aws.amazon.com/documentation/>

4.2 Tooling

- Atom Editor
- Rbenv Ruby environment manager
- XCode IDE for iOS Development
- Android Studio IDE for Android Development

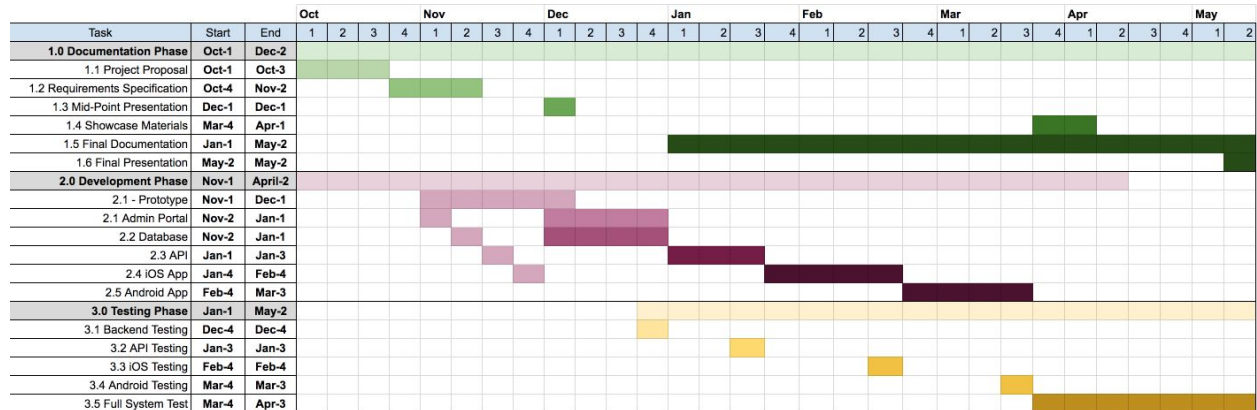
4.3 Hardware

- Development machine - Apple Macbook Pro 15
- iPhone 6 & iPhone 7 test devices
- Samsung Galaxy S6 test device

5. Project Plan

The project will be split into three main phases: the documentation phase, the development phase, and the testing phase. According to the deliverable dates set out by the NCI faculty, there will be some overlap in these phases. See item 5.1 - *gantt chart* for the exact spread throughout the project.

5.1 Gantt Chart



6. Technical Details

6.1 - Backend Platform

Implementation Language	Ruby, HTML (Hamli), CSS (SCSS), Javascript
Deployment	Amazon Web Services Elastic Beanstalk
Libraries Used	Ruby on Rails, Various RubyGems as per requirements, Bootstrap, RSpec test framework

6.2 - API

Implementation Language	Ruby, serving JSON over REST interface
Deployment	Amazon Web Services Elastic Beanstalk
Libraries Used	Ruby on Rails, Various RubyGems as per requirements, RSpec test framework

6.3 - Database

Implementation Language	mysql
Deployment	Amazon Web Services RDS

Libraries Used	TBD
-----------------------	-----

6.4 - iOS Application

Implementation Language	Swift
Deployment	iOS App Store via XCode, Fastlane build scripts
Libraries Used	CocoaTouch Framework

6.5 - Android Application

Implementation Language	Java
Deployment	Google Play Store, Fastlane build scripts
Libraries Used	TBD

7. Evaluation

The finished product will be evaluated with respect to various criteria.

7.1 - Unit Testing

TDD - Test Driven Development will be employed when writing all features. In this way, tests will be created before functionality is built. This will serve to provide a specification for each function, while also giving us maximum code coverage. Tests will be run on integration to ensure all features are bug-free.

7.2 - Manual Testing

The final week of each development phase will be used to conduct manual testing on each feature. This week will be used to identify and fix any bugs introduced during that development period. Appointed alpha testers will be asked to evaluate the platform at certain points during the development cycle. In the case of the mobile applications, the Crashlytics mobile testing framework will be used to deploy test builds to testers own devices for evaluation.

7.3 - Requirements Evaluation

The entire product will be evaluated in conjunction with the requirements specification. Each requirement will be evaluated individually to determine if the final product fulfils the requirement correctly.

Appendix 2 - Monthly Journals

Reflective Journal

Name: Niall Quinn
Programme: BSCHE SD
Student Number: X13108727
Month: September

Achievements

This month I brought my Software Project idea from the idea stage to putting a tentative plan down on paper, and pitching to the panel of Michael Bradford, Manuel Tova-Izquierdo and Joe Molumby.

I fleshed out the core pieces which I want to be present in the final project, and settled on an initial plan for which technologies I would like to use. I decided to focus solely on the vertical of Motor Racing Drivers.

Reflection

I am happy with how my idea was received by the panel and I was happy to have my pitch accepted. They offered some advice to me on the best areas on which to focus in order to maximise the potential of my project. I will be taking these on board. I am now excited to build upon it more and get my plans down on paper in the form of my Project Proposal, and later my Requirements Documentation. I am happy with my idea as a whole, and I feel passion for the project which will carry me through the 6+ month development period.

Improvements

I have been working hard this college year to manage my time better than in previous years. At the moment I am juggling a full time job where I am Senior iOS Developer bringing a large project to market in Q4, with a pretty intense workload for year 4 in this degree. I have decided to use Asana to track my deliverables, and the upside of this is that I get a deliverable calendar, along with notifications when deadlines are approaching. I believe that this approach will help me to deliver every module well and maximise my potential to score high marks.

Reflective Journal

Name: Niall Quinn
Programme: BSCHE SD
Student Number: X13108727
Month: October

Achievements

This month I completed the Project proposal deliverable and the Requirements Specification.

Reflection

I am happy with how this stage has gone. These were quite big documents and finding time to complete them has been difficult. However, I am very happy to have worked through the requirements specification putting a lot of thought into the implementation of my project. I feel very ready now to being the coding phase.

Improvements

Again, time management is the hard part this year. With a full time job at a growing startup, it is not easy to juggle all. I am overall happy with my time management but of course this could be improved.

Reflective Journal

Name: Niall Quinn
Programme: BSCHE SD
Student Number: X13108727
Month: November

Achievements

During November, I began to work on the codebase of the application. I am very happy with the progress made during this month, as I now have a working application. It is now possible to create an Admin user, and log in as this user. It is now possible for an Admin user to create a customer. It is possible for a Customer to log in. The customer is greeted with a view which lists all of the news items for the customer. These items are sandboxed as per the multi-tenancy requirement. The customer admin can add, edit and delete news items. The news items are created using a WYSIWYG editor..

Next Steps

I will continue to flesh out the code base, working towards a working prototype for my interim presentation.

Issues

I have not yet met with my project supervisor, Pdraig deBurca. We had scheduled to meet initially, and I had to cancel a few hours before the meeting due to work commitments. We scheduled to meet the following Monday at 5.30. I arrived 10 minutes before the scheduled time and Pdraig was not in the Associate faculty room. I emailed Pdraig and waited until 7pm, but left at that point with no reply. Pdraig later emailed me apologising that we had 'missed each other' and we set about rescheduling. Since this contact I have not received a response from Pdraig to my emails. Another student has told me Pdraig is ill, but I have not received this information first hand

Reflective Journal

Name: Niall Quinn
Programme: BSCHE SD
Student Number: X13108727
Month: December

Achievements

I presented my mid term presentation to Pdraig de Burca and Eugene McLaughlin. I was very happy with how the presentation went, and I feel I did well to show the potential of my idea and how much progress I have made to date. I was awarded a mark of 21.75/25, which I am very happy with. I met with my supervisor Pdraig de Burca and we spoke about my presentation and my plans for development.

Next Steps

I have exams coming up so I will be shifting focus to those for the time being. I will be keeping my mind on my project and thinking about solutions to the various technical challenges that lie ahead.

Issues

I encountered no issues in December.

Reflective Journal

Name: Niall Quinn
Programme: BSCHE SD
Student Number: X13108727
Month: January

Achievements

January was a tough month for me. I fell ill the week after christmas which severely disrupted my work and study for exams.

Next Steps

February will mark the return to heavy work on my project. I have a lot to do to catch up on my project plan.

Issues

Due to my illness I had to spend some time seeking a deferral of the christmas sitting exams. This caused some stress but all was resolved thankfully.

Reflective Journal

Name: Niall Quinn
Programme: BSCHE SD
Student Number: X13108727
Month: February

Achievements

In february I implemented the Customer and introduced multi tenancy to the platform. I worked on the styling of the entire platform and got closer to the final look and feel that I was after. I began work on the iOS and Android designs before the real work begins on those platforms. I implemented the image uploader for the post. I met with my supervisor Pdraig De Burca. We spoke about my progress and if my project plan was moving as expected.

Next Steps

In March I plan to implement the Theme, Bio and Calendar items. These pieces should be quicker to develop as I am learning rails and feel more comfortable as I go along.

Issues

I ran into a few bugs during this period.

Reflective Journal

Name: Niall Quinn
Programme: BSCHE SD
Student Number: X13108727
Month: March

Achievements

In March I added the functionality to edit the theme for the customer. This was a piece that was daunting to me, but I was pleasantly surprised when my solution worked as expected. I began work on the mobile applications.

I scheduled an appointment to meet with my supervisor, Pdraig de Burca. However he was ill and we rescheduled for April.

Next Steps

In April my plan is to flesh out the mobile applications fully. I am more experienced with mobile applications so I expect this will go quite smoothly. The only possible stumbling block will be integrating the theme in a satisfactory way.

Issues

Again, there were no real issues to report in March.

Reflective Journal

Name: Niall Quinn
Programme: BSCHE SD
Student Number: X13108727
Month: April

Achievements

In April I completed the functionality in both mobile applications and fully integrated them with the API. I prepared applications for testing and delivered to testers. I used a google form to gather the responses from the testers. I met with my supervisor, I had some questions about my project and my mind was put to ease. I am feeling good about my progress. The application is now deployed live on the internet and working as expected.

Next Steps

In May I need to finish the technical document and prepare my presentation.

Issues

Reflective Journal

Name: Niall Quinn
Programme: BSCHE SD
Student Number: X13108727
Month: May

Achievements

I finished the technical document and presentation slides. I finished the final pieces of the product in the codebase and deployed the final version to Heroku.

Next Steps

Wait for results :)

Issues

Appendix 3 - Usability Questionnaire

This questionnaire was posed to the testers in the form of a Google Form. I have extracted the questions to text form for easier viewing. The form can be viewed at:

<https://goo.gl/forms/IKWh2KQfD2sjGCU72>. Results can be seen in section 4.3

RFBBase Usability Testing

RFBBase is a marketing platform for racing drivers. The product consists of a backend platform for adding and managing content, and iOS and Android applications for users to consume the content. The types of content include:

- News Item
 - Title, body, Header Image and Square Image
- Bio
 - Driver Info
 - Social and website links
 - Career details
- Calendar
 - Events with Name, Date, Description and Image (Track Map)
- Media
 - Image + Caption

The mobile apps are white labeled, and implement the theme which is configurable on the backend platform. The theme allows the user to choose the Primary, Inverse Primary, Primary Text and Secondary Text colours.

Thank you for taking the time to use the product and thanks in advance for your feedback. Feel free to play around and change some content and view that in the mobile application. Please bear in mind that there are other testers in this session, so please don't delete everything.

Your Info

Name:

Gender:

Profession:

Company you work for:

Age Range:

What is your primary device used for consuming online information:

Platform

Was it immediately obvious what type of content was presented to you on landing?

Click around, take some time to navigate around the platform. You can:

- Add, Edit, Remove News items
- Add, Remove Media Items
- Modify the Driver Bio
- Add, Edit, Remove Calendar Items
- Modify the App Theme

Read the following statements and on a scale of 1 to 5 mark how much you agree. (1 I do not agree, 5 I fully agree)

The site navigation is effective

It is clear which Items on the platform on which I can perform a DELETE operation

It is clear which Items on the platform on which I can perform an EDIT operation

It is clear which Items on the platform on which I can view detail

The WYSIWYG (What you see is what you get) editor is a good addition for styling text in News and Bio

I find the site aesthetically pleasing

Feedback - Please supply any feedback

Mobile Application

The mobile app exposes all of the content created on the backend platform to the fans.

It is clear which sections are available and how to navigate to each section

It is clear how to show the detail for a news item

The social links on the bio page are a good addition

The layout of the media page is user friendly.

I find the app aesthetically pleasing

The theming engine is a good addition to the application

Feedback - Please supply any feedback

Appendix 4 - Showcase Poster



Niall Quinn
BSHCE Computing



One Place For Your Fanbase

RFBASE is a marketing platform for the modern Race Driver. Reach your fanbase through your personalized iOS and Android Applications, right from the App Store!

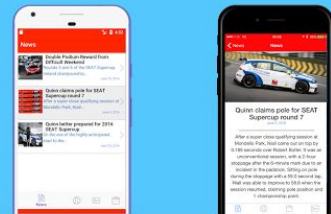
YOUR App In The App Store And Google Play Store



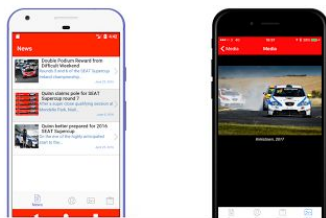
Add Content Using Our Secure Admin Platform.



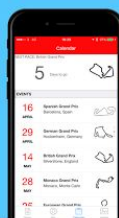
Deliver your latest news with scheduled delivery.



Kepp Your Fans Interested With Rich Media.



Fans Will Never Miss A Race - Race Calendar With Countdown



Apply YOUR Brand Using Our Theme Engine



TECHNOLOGIES



iOS



ANDROID



PostgreSQL

