City, University of London

BSc Computer Science

Academic year: 2021-2022

ProtYes

A mobile application that manages the Protests in your area

Author: Suad Warsame

Suad.warsame@city.ac.uk

Consultant: Giacomo Tarroni

Table of Contents

Abstract

In recent years, there have been widespread protest and demonstrations held on extensive scales across the globe. The details of these events have been extracted from social media posts and word of mouth. In this project, the author will design and create a prototype that enables the users to learn more about upcoming protests in their area. The proposed project is titled 'Protyes' and aims to enable researchers, active protesters, academic students to keep track and educate on themselves on future protests. It will clearly outline the date, time, and place of the planned events so users can utilise the information provided to successfully attend the event. The project will contain a clear and concise framework of the stages undertaken to create the final IOS application.

Chapter 1: Introduction

1.1 Problem to be solved

The focus of this project is to develop an interactive platform that enables the ability for users to track and be updated with any demonstrations and protests or riots that are happening in their area, as well as interacting globally. The application is simple, it is essentially a global protest tracker whereby the organiser of the protest can log into an admin account within the platform and then make an event for their planned demonstration or protest, provided they lawfully have written to the authority to proceed. The application will specify if it is a protest or if it is riot. Riots are more unpredictable, so there would not be as many indicators on the map for it. A recent example of a sudden riot is the South Africa Riots in July 2021, whereby more than 300 deaths were recorded (Harding, 2022). At the time of the riots, communities would get together to prevent looting and further political unrest, had an app that clearly indicated where the looting and riots were taking place, it would be easier and quicker to target. This application aims to allow registered users to give updates which will then be verified by fellow users to confirm its legitimacy. The application will be designed to solely focus on the events of demonstrations and protests happening, taking into consideration accessibility, confidentiality, and availability of information. Demonstrations and protests are prevalent and could happen at any time. Therefore, the planned demonstrations and protests could be updated on the application, meaning anyone looking to join these or even avoid traffic at a certain area would benefit. Usually, people would search on social media networks such as Facebook and Twitter, but the idea of this project is to integrate this information into a live interactive map.

1.2 Project Objectives

The main objective of the application was to provide a service that enabled the users to actively add upcoming events such as protests and demonstrations. This project has an interactive map which used Apple Maps API and integrated it with a Next Events page, clearly explaining the events in depth which is supported by visuals. The Apple Maps APIs would be used to connect and fetch content for the map interface utilising the open-source components to mirror these on the application. It will be global, so information regarding different countries would be available that is submitted and approved will be reflected on the map. Research and background information on the topic will be conducted to understand the topic in detail. The sub-objectives that will be followed throughout this project are the following:

1.2.1 Sub-Objective 1: Login screen

- Let user chose between the option of creating a new account or log back into registered account
- Enable user to fill in form to register account if previous account is not attained

1.2.2 Sub-Objective 2: Home page

- Incorporate a friendly user interface using Apple Maps API to implement this

1.2.3 Sub-Objective 3: Next Events tab

- Provide a tab that has all the upcoming events that have been submitted through the app

1.2.4 Sub-Objective 4: Admin page

- Provide a tab that provides the admin to essentially approve and update the Map and Next Events page

1.2.4 Sub-Objective 5: Profile page

- Enables admin user to view the submitted protests by the users, and enables all users to change their profile picture and check account details

1.2.6 Tools

The following tools are going to utilised to achieve my objectives:

- Utilising the language Swift: It is a language developed by Apple and this would be applied to the frontend and backend coding
- Applying Apple Maps API: This is the main feature of the application, where the JavaScript allows the customisation of the map with personal content and imagery to display on the proposed mobile devices

- Coding on XCode: Throughout the project, the platform XCode is used to code using the language Swift. The setup will require the Bundle ID which is connected to a Google Cloud Account for Firebase and there onwards Swift is utilised

- Use of Cocoa Pods: It will manage library dependencies for the XCode projects, where the projects are specified in a single text file called PodFile, which will be created

- Documenting on Firebase: Any information saved is going to use Firebase, which provides storing, tracking, and reporting tools, showing it live. It is a database that is stored on Google Cloud

- Creating a Wireframe: This was useful in creating the design of the application. FluidUI was used to sketch the user interface

1.3 Project Beneficiaries

The project will be beneficial in reducing the collateral damage riots leave as users will be more aware of the riots in their area. It will allow users to have a live update of protests they can join which promotes freedom of speech. It will greatly benefit the local authorities as they will have the ability to utilise their resources effectively for there to be an equal balance in their priorities. It allows the event organisers to plan their events in a professional manner, setting the time, date, and place for their occasion. Academic students and researchers that have interests in field of political movements can also learn from this application.

1.4 Work performed

1.4.1 Analysis phase

During this phase, the author highlighted the framework of the project by including the functional and non-functional requirements of a system. These are represented using Volere system specifications in Appendix D to test the criteria and that the solution provided during this stage fits the requirement. Questionnaires have also been utilised during this stage as it assisted the author with collecting data on Protyes. This is later discussed and found in Appendix E.

1.4.2 Design phase

The author sketched a basic interface on paper before proceeding to create a storyboard on FluidUI. An example of this lo-fidelity wireframe which provided the blueprint for the application is in Appendix G. For a more real-life and advanced version, a high-fidelity wireframe was created using FluidUI and is discussed further in chapter 5. For further design work, several Use Case, Sequence,

Cloud Architecture, Navigation Architecture and Entity Relationship diagrams were created at the beginning of projected as well as during the process to establish that the set requirement was met.

1.4.3 Implementation phase

Using XCode, an IOS app was created by the author whereby an Apple Map API was used and synced to a Firebase Real-time database. Once the Google database was successfully set up, the author proceeded to coding using the Swift language on XCode, amending the code if there were present bugs.

1.5 Assumptions

The author had no prior knowledge in developing an application or coding in Swift. Majority of the research conducted was using YouTube videos and solutions on Stack Overflow to become familiar with the environment. Due to the extent of the project, the author also assumed that the coding element was so be given higher priority, but this project was only a prototype.

Chapter 2: Outputs summary

In this chapter, the author explains the outputs of the project which include the intended outputs, the recipients, beneficiaries, and link to appendices.

2.1 IOS Application

Description	The first output is a software application called 'Protyes' which allows the user to manage protests near their area
Output	The project will produce a compiled application that can be uploaded to the App Store and thus downloaded by any Apple users. Swift language will be utilised on XCode. It will have several classes and a few lines of codes which were written by external sources and adapted/tailored by the author. The author has 1670 lines of authentic code for the entire project, which were written by the author, not including comments.
Type	IOS Application
Recipient	Author, Consultant, Reader
Benefits	Using XCode, the author will have the ability to code an application that appeals to protesters, the government official and academics
Link	Code submitted on Moodle

Table 1: IOS Application Output

2.2 Firebase Database

Description	This output configures the Firebase real-time database to XCode. A working connection is then established from the Google Firebase Database to XCode enabling user authentication
Output	The project uses the Google Firebase Database to not only store data, but it also supports any authentication for the system. Having created a Google account, the author proceeds to sync the 'plist' file from the real-time data to XCode.
Type	Google Firebase
Recipient	Author, Consultant, Reader
Benefits	Any information saved is here and allows you to store it and show it live
Link	Appendix H

Table 2: Firebase Database Output

2.3 Wireframes

Description	Utilising FluidUI to create a wireframe to display the functionality of each tab/screen on 'Protyes'
Output	This project uses FluidUI during the design stage to create a Wireframe for each screen of the application. This is followed by the sketch on paper visualising how the app will look like. The storyboard consists of 14 screen sketches
Type	FluidUI
Recipient	Author, Consultant, Reader
Benefits	Will help the author visualise what the application will look like and help with the smooth development and implementation of the project
Link	Appendix G

Table 3: Wireframes Output

2.4 Test Cases

Description	This output provides all the use case tests used in this project
Output	It ensures that all the main functionalities have been tested and then recorded. By doing this, the author can check that the application is working to the expected quality
Type	Word document
Recipient	Author, Consultant, Reader
Benefits	The author will have the ability to test the main features and make changes if necessary
Link	Appendix E

Table 4: Test Cases Output

2.5 Questionnaire

Description	This output will produce a document that will record the users experience using the 'Protyes' app
Output	A document consisting of 11 questions given to a sample size of 8 people, **one** of which regularly attends protests. This research tool is used by the author to gather data from the target audience of academics, researchers, and protesters. The questionnaire will include both open and closed ended questions to collect as much as possible
Type	Word document
Recipient	Reader
Benefits	The questionnaire will help the author understand the opinions and thoughts of the user and have a better insight into the user's experience
Link	Appendix E

Table 5: Questionnaire Output

2.6 Mobile Application Demonstration

Description	The project will be supplied with a video to highlight how to navigate around the application 'Protyes'
Output	The video is 14 minutes long to ensure that it is thorough in its explanation so the users can easily navigate through the app. There is a YouTube link that would be provided
Type	Video
Recipient	Author, Consultant, Reader
Benefits	This will allow the user to effectively understand how to use the app to its full potential and navigate from tab to tab, explaining both the obscure and obvious through simple visuals
Link	YouTube Link, Moodle submission

Table 6: Mobile Application Demonstration Output

Chapter 3: Literature review

This chapter will discuss the tools and software the author utilised as well as conducting research on similar applications to Protyes.

3.1 Analysis of similar/existing applications

Through extensive research, the following applications were discovered to have similar features or purposed to the proposed project Protyes.

3.1.1 Citizen

Citizen is a free app to use that enables users to connect to real time 911 alerts. The platform is available on both IOS and Android and can be downloaded from their respective stores. The mission of Citizen is to create a safe world environment, whereby "transparency that bonds and that empowers everyone in a community, from city council to residents" (Citizen: Keeping you safe & informed, 2022) is prevalent. Citizen intertwines a connection with the individual, city officials, local communities, and emergency responses with the idea that word can spread faster if all parties communicate. The application supports the Map feature whereby there are live points annotated on the map and the uses can upload videos of the situation and others using the platform can also react with the content.

3.1.2 Causes (previously known as Countable)

Causes is an application that allows users to make informed decisions whereby they aim to keep "educated about the news that they're consuming and discover more modernized ways they can get involved" (Meet Countable: The Non-Partisan Political App You Need to Download, 2022). The main purpose of the app is to feed political information to their users, and it is up to them to make their informed decisions. Essentially, it is an informative application that promotes political awareness.

3.2 Tools and software used

During the project, the following applications were used by the author:

3.2.1 XCode

Regarding the design phase of the project, Swift is the most suitable language and uses XCode, which is "an Integrated Development Environment for constructing applications for Apple platforms like iPhones, iPad, MacBooks, etc" (Aggarwal, 2022). This approach will enable the ability to build an

IOS app, test it out and debug any errors. The approach allows the programmer to work on the visual elements of the application via the interface builder before integrating it with the code.

3.2.2 Swift

Swift was developed by Apple Inc. for iOS and is a multi-paradigm programming language. “Swift code is compiled and optimized to get the most out of modern hardware” (About Swift, 2022). This language is leveraged by SwiftUI which offers support to design features. Using this language, developers can use concise syntax to create fully featured apps.

3.2.3 Firebase

The Google Firebase Realtime Database was implemented in this project and “provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and product experiment” (Rosencrance, 2022). The data stored here is synced across all the clients in real time and still will be available when the app goes offline. It is a cloud-hosted NoSQL database allowing the users to store and sync in real time.

3.2.4 YouTube

YouTube is a video sharing platform that allows the users to discover a variety of content. The author viewed the content videos regarding Swift for the sole purpose of learning how to code in the language. The material that was provided on these videos was then applied onto XCode. The reference section indicates which videos were used during the process.

3.2.5 Stack Overflow

A vital instrument used in this project was stack overflow to target the errors encountered during the coding stage on XCode. Common errors the author faced would result in Stack Overflow being utilised where step by step guides were provided on a forum.

3.2.6 Apple Developers

“Apple Developers is Apple Inc.'s website for software development tools, application programming interfaces (APIs), and technical resources” (Apple Developer - Wikipedia, 2022). This approach includes the support page which provides resources. The author utilises Developers to research a variety of documentation to help with configuration and coding.

3.2.7 Word Document

To documenting and recording the authors progress, Word document was utilised and as a means for user testers such as questionnaires. It is a word processor used for documentations throughout the project and the author utilised this for aspects such as PDD, Final project and questionnaires.

3.2.8 Conclusion

Having conducted research on similar products that serve a similar purpose to Protyes, the author concluded that a prototype of a mobile application is something new to the market and was to be created where the sole purpose was to track upcoming protests. By utilising Google Firebase, this could be successfully achieved as it provides a platform for real time data to be fetched and saved onto the application.

Chapter 4: Methods

4.1 Methodology

As mentioned previously, this project will utilise the agile software methodology but mainly focus on the SCRUM framework. The agile development paradigm aims to react quickly to changes in customer's needs (Petersen, 2010). With this project solely revolving around the end users' requirements and needs, it is imperative that there is constant revision of different stages of the project. Agile software development enables developers to go through a cycle of "planning, executing, and evaluating" (What Is Agile Methodology in Project Management?, 2022).

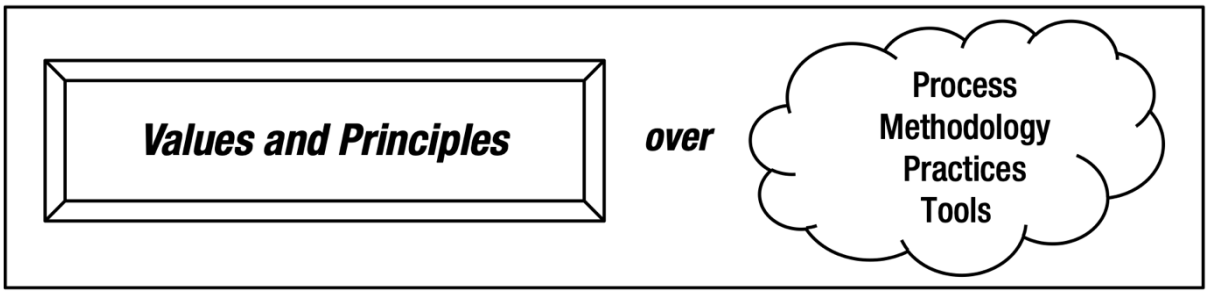


Figure 1: Agile demonstration: Agility is more attitude than process, more environment than methodology (Highsmith, 2004).

4.1.1 Scrum

Throughout this project, the SCRUM framework has been applied to develop and sustain the complexity of the product, clarifying the value of the development practices so that the author could improve. SCRUM is based on empiricism, which is a theory that knowledge is constructed on experience derived from what is known (Schwaber and Sutherland, 2016). Revision of the project is imperative as constant reviews would be used to tailor an improvement for the current framework. An incremental and iterative approach is what SCRUM encompasses and what the author adopts to each stage of the project using sprints. The sprints were split into 4 which lasted around a month. Prior to starting the sprint, planning of the task had to be assembled which included the requirements that had to be met and at when. Following the planning stage, the design stage followed then the implementation stage is finally applied. The designs essentially laid out the development framework for the requirements within the sprint. The author compiled records to document the progress made throughout the project which will in turn contribute to any future adaptations.

4.1 Design

This section will compromise the high-level contents the author planned before developing the application on XCode.

4.1.1 Wireframes

In order to grasp an understanding of how the website was going to be designed, the author created high-fidelity wireframes to help with visuals. This allowed the author to design the user interface which in turn created the frameworks for the functional elements of the application.

4.1.2 Navigation Architecture

Having completed the wireframes, the author utilised each screen to create a sequence of navigation from one screen to another. This ensured that there was a constant and predictable user experience that paved the way for how the application was to be designed.

4.1.3 Use Case Diagram

This UML model was produced at the project's planning stage, and it outlined the system's users and their interactions with it. The system users are the actors, and this diagram serves the purpose of representing a high-level overview of the relationship between the system, actors and use cases.

4.1.4 Use Case Specification

Following the creation of the Use Case Diagrams, the author was able to compile a documented description of the functionality produced by the system and the actor’s interaction. Use Case Specification essentially captures the actor-system interaction and is a high-level statement and the author utilised this to sculpt how the system should behave.

4.1.5 Sequence Diagram

A sequence diagram is an interactive graphic that depicts the order in which messages are sent between objects. This helped the author comprehend the needs for a new system.

4.1.6 Class Diagram

Class diagram is an outline of the system and is used in design to model the components that contribute to the structure of the system. It consists of classes which contain objects that inherit other classes. The author utilised this to have an overview of the system.

4.1.7 Entity Relationship Diagram

The author utilised this graphical representation of the system which is important in this project as the author was going to store data in a database, and the Entity Relationship Diagram allows the modelling of the data. It allows a visual representation of how the data can be structured which in turn is useful for the author to comprehend what the project is going to be.

4.1.8 Cloud Architecture Diagram

This diagram visually represents the parts of the system that utilise the cloud. Google Firebase is applied, and the author uses this to see how the components work with each other.

4.2 Implementation

4.2.1 Tools & Software

During the development of Protyes, there were multiple tools and software that were operated, and it is highlighted in this table:

Tools used	Purpose
GitHub	This software development tool enabled the author to monitor the progress of the app as it is a version control platform. All the project code that was conducted every few days was pushed onto repositories. This ensured seamless uploads of regular edits of the project on the software.
XCode	An Integrated Development Environment which is used for developing software for macOS. The application is appropriate for multiple Apple platforms and the author created an IOS app using the language Swift.
Microsoft Word	Word processor used for documentations throughout the project such as PDD, Final project and questionnaires.
Microsoft Excel	A platform for tracking progress and utilising spreadsheets to organise and manage development in the project. The author referred to the initial Gantt Chart where a definitive overview of the project's timeline was created, and deadlines were set.
FluidUI	A HTML design tool which essentially is a mobile interface utilised by the author to help with the user interface and wireframes for the IOS application.
Visual Paradigm	This software development tool is designed to assist with UML modelling and the author utilised it to create Use Case, Sequence and Entity diagrams.

Table 7: Tools and Software used

4.3 Sprints layout:

This project is going to be structured into sprints, specifically broken down into four parts. The aim of each sprint will be to produce a functioning IOS application where each one will contain specific requirements.

4.3.1 Sprint 1: Foundations

The title of the initial sprint is 'Foundations' where this aims to showcase the development of the basis of the application.

4.3.1.1 Connect Firebase to Swift:

Having conducted research on an easy and reliable database, mentioned previously in section 3.2.3, Google Firebase was chosen. This process of connecting Google Firebase to Swift ensured that data was stored and synchronised in real time using a NoSQL database (Firebase Realtime Database | Store and sync data in real time, 2022). The steps taken to set up the database were effortless. A guide provided on the Google Firebase website, which could be accessed at any time, was read prior to the set-up as well as a step-by-step tutorial when proceeding with the connection. Majority of the project required this connection to the database, and it was imperative the connection was successfully done. With the assistance of the Google Video (Getting started with Firebase on iOS - Firecasts, 2019) provided on the Firebase website, the author was able to grasp the understanding of how to register the app to the Firebase console using the 'bundle ID', how to add SDK repositories and how to download the configuration file onto the XCode project. This ensures that any information that was entered through the app was presented through the live data in the Firebase database.

4.3.1.2 Firebase Authentication:

In order to incorporate the authentication, a document provided by Google on the Firebase webpage was utilised. The document titled 'Get Started with Firebase Authentication on Apple Platforms' specifically outlined how the users could sign into the application using email address and passwords and how to include the SDK files (Get Started with Firebase Authentication on Apple Platforms | Firebase Documentation, 2022). Once this was set up, the author continued to build on the code retrieved from the Google Firebase document, heightening the functionality for storing the user's information in real time. The project incorporated three main authentication features: sign-up, sign in and sign out (once on the profile). Within the sign-up page, there is an option to sign up or sign in as a standard user or admin. Once the user is on the chosen profile, there will be a profile tab where they can locate the sign out button.

4.3.1.3 Firebase Storage:

Following from Firebase authentication, it is needed to achieve any actions on the Cloud Bucket's data and files, which is the default set up. This meant that for this project, as it is a prototype, the functionality of the user generated content is not public and in fact hard coded, so only a small number of images have been stored by the author. In other words, the user must sign in to access the ability to store data. Incorporating this storage feature meant that the author referred to the 'Get started with Cloud Storage on Apple platforms' document provided on Google Firebase in order to create working Cloud storage buckets (Get started with Cloud Storage on Apple platforms | Firebase Documentation, 2022). These buckets essentially provided an environment to securely upload chosen content. To do so, packages had to be added onto XCode and the Firebase SDK repository was then added whereby a Cloud library file was chosen.

4.3.1.4 Firebase Firestore:

Considering all the files and data that needs to be stored during this process, Firebase was the solution to containing and syncing these files. Firebase revolves around the concept of storing data in documents which are further organised into collections (Data model | Firestore | Google Cloud, 2022). These documents include 'key-value' pairs which essentially is a piece of data that is identified to a random name, and all these must be stored in collections. For the project, the author used 4 collections and multiple documents within these collections. The data that was utilised in this project was manually inputted for example, the images, and it essentially laid out the framework for the architectural design.

4.3.1.5 Navigation:

Finally, a navigation feature had to be incorporated to allow movement from one tab to another. This component was imperative as the destinations had a fixed position throughout the different screens and being ergonomically placed meant that it was user friendly and could be reached easily on a mobile device. In terms of coding this on the app, the author created multiple Swift UIViews for the different screens that were projected.

4.3.2 Sprint 2: Map Integration/ Admin Features

The second sprint was designed to include all the components that interacted with the map and functions mediated by the admins.

4.3.2.1 Implement Apple Map:

The first plan of this project entailed that Google Maps API was going to be used but after careful consideration and discussion with consultant, Apple MapKits which was already integrated in Swift was used. Using the MapKit framework, it was easy to build an interactive map that also stored its data on Firebase. A YouTube video clearly demonstrated how to use 'MKannotations' to include information that was coded onto the map (Custom Annotations for Maps in SwiftUI, 2021). By utilising the MapKit feature, the author was able to inherit a framework that embedded an interactive map with the ability to determine where to place annotations for point of interests. The author implemented fixed coordinates so when the user logs in, they are automatically zoomed into one location. In future updates, there are plans to make it locational, meaning that depending on what location the user is in, the map will be revolved around that area.

4.3.2.2 Implement Live Protest:

As the one of the prime driving forces of this project, the implementation of the live protests on the application was domineering. In line with the project's objectives, the author included a tab whereby standard users could write any upcoming events in the given fields and the admin would be able to view these and when the event is live add it on the interactive map as an annotation. This allowed the function to regularly updating the database, and monitoring contents that were determined as a new protest location.

4.3.2.3 Implement Admin Feature:

The main functionality of the app is to alert people of protests in the area through the submission of future protests on the app. The application contains a screen whereby regular users will be able to submit details for any upcoming protests in the area which is then managed by the admins who use the application. When the admin verifies the protest, they will have the ability to call a point of interest on the map. With the help of the YouTube video that explained how to fetch data from Firebase and display it in XCode by the Firebase YouTube channel, the author was able to achieve a framework that enabled the admin to make changes on the app in real time (SwiftUI: Fetching data from Firestore in real-time, 2020). As well as having the capability to create an annotation on the app, the admin can amend information direct from the database and display it using Swift UI.

4.3.2.4 Implement Next events:

Additionally, the admin will have the ability to control the back-end functionalities, that is from submitting a protest as an annotation on the map or listing an upcoming event on the Next event page. However, the Next Events screen is available for both a standard user and an admin user. Information

was entered by the users and data was then extracted from the database. This was developed using a YouTube video that was adapted to fit the requirements of the Next events page (SwiftUI Simple Blog App, 2021).

4.3.3 Sprint 3: User Interaction

This sprint laid the outline for the prime functionalities that was operated by all users.

4.3.3.1 Login page

As soon as the application loads on the user's mobile device, there will be two buttons to choose from: 'Get started' and 'Admin'. Upon clicking the 'Get started button', the user would be prompted to log in or create an account. This integrated email authentication from Firebase Authentication allowing a free and easy authentication setup. Once the details are entered, there are checks completed in the background, for instance, a user cannot create an account if the password is not six characters long. The author applied an online video tutorial to execute this feature (SwiftUI Firebase Chat 07: Sign Out of Firebase, 2021).

4.3.3.2 Profile page

Upon logging into the application successfully, the user (both standard and admin) will have the ability to navigate to their profile which is situated as the last tab button. Both users will have the ability to sign out of their account, but the admin will have an added feature of accessing and editing the contents of the database. This was created so that future admins using this prototype, do not have access to the Firebase account. It makes the content of the firebase readily available without having go compromise any Firebase credentials that is owned by the author.

4.3.3.3 Protest submissions

To do this the user can click on the screen where there are fields to scribe details of an upcoming protest. Having built the code from watching a YouTube video called 'Fetch and save data', the author had the ability to tailor and adapt the code to fit the requirement to submit protest details (SwiftUI with Firebase - Fetch and Save Data, 2021). Once the details are entered, they will fall under the ProtestSubmission document in Firestore database, and the admin will have the ability to extract this.

4.3.3.4 Display Protest submission

To display the content of the protests that were submitted, the author utilised a YouTube video then extend and adapted the code to fit the criteria set (Integrating MapKit with SwiftUI – Bucket List SwiftUI Tutorial 4/12, 2021). The protests had to be displayed on the Map screen but the way to plot

it was through a submission by a user (both standard and admin), then a destination point would be nailed.

4.3.4 Sprint 4: Finalisation

This sprint offered an environment for the app's core components to be improved, as well as a chance for the author to address any defects.

4.3.4.1. Implementing Protest Route

From the basic concept of single points on the map, the author wanted to improve this by integrating a route like feature for the protest path. This would have offered a more complex user interface, allowing the user to easily observe the protest's exact course. To do so, the author utilised and adapted the code retrieved from a video and implemented that into XCode (SwiftUI Tutorial - MapKit, Route, and Directions, 2020).

4.3.4.2 Fixing any Errors

The author ensured that the final submission of the project had no faults in the code in order to generate a competent version of Protyes. The author was able to make frequent updates and solve any mistakes that happened along the route courtesy to version control and the use of GitHub.

4.3.4.3 Improving User Interface

After finishing the application's foundations, the author focused on improving the user interface. Sprints 1-3 laid the groundwork for a working software, after which the author improved the application's user interface. Because this is the initial version of Protyes, the author made no major modifications and instead focused on making it more user-friendly and customizable. The author concentrated on features such as the size of the text boxes, the names of the text fields, and the success/error message displayed to the user.

4.4 Analysis

4.4.1 Functional and Non-Functional requirements

The requirements were separated into two groups: functional and non-functional. Functional is essentially the functions the system does or doesn't perform, and non-functional is how the function is carried out. To further explain this, the author utilised the Volere template whereby a clear outline the two requirements we composed. Volere templates display more details regarding each demand, such as the ID, kind, a brief description, client satisfaction and discontent, and so on (Robertson, 2012).

4.4.2 Questionnaires

The interview questions intended to understand how often people attended Protests and if they knew an entry point to gather their data on upcoming protest. The questionnaires also go into further depth about how the project might be enhanced in comparison to previous initiatives and ways of gathering protest data. These could be found in Appendix E**.**

4.5 Evaluation

4.5.1 Testing

As the project followed the Agile method, specifically SCRUM, it was imperative that all framework was tested periodically. One form of testing was conducting User Acceptance Tests (UAT) after each sprint. This is in Appendix E**.** The UAT verifies that the functionalities implemented in each step are performing properly and are not producing any serious defects. It's critical to test during each sprint since a bug in one sprint might trigger issues in subsequent sprints. First and foremost, critical components and foundations of the programme were emphasised. When a function failed the UAT, the author attempted to resolve the problem; however, if the author ran out of time and the function was not high priority, the author returned to it after finishing other sprints. Other tests included Test Cases and Usability Testing which is further discussed in section 5.4.1.

4.5.2 Reused code/ Tutorials

For the duration of the project, the author utilised multiple online tutorials and documentation to support the development. All the video tutorials used were adapted and tailored to fit the author's needs. The tutorials and documentation utilised are mentioned in section 4.3-4.5 and are listed in the references.

Chapter 5: Results

This chapter will cover all the results constructed in the project. It includes detailed requirements, software architecture, specific algorithms and coding decisions, verification and evaluation results.

5.1 Analysis

5.1.1 Questionnaires

To have a good understanding of the intended target audience and accessibility of Protyes, the author conducted questionnaires. The structure of this questionnaire covered a variety of questions from the user's interaction with this prototype to their thoughts on it. The result of the questionnaire is in Appendix E**.** From the questionnaire, the author understood that the main concern was the inability to automatically add coordinates, which is feedback the author took on board and plans on implementing in future updates. The interactive map was proven to be most popular with the testers.

5.1.2 Prioritisation of requirements

The author produced a list of requirements premised on the data received from the questionnaires and similar application analysis. Following that, the author determined which functions were functional and which were non-functional. The author created a prioritisation method to determine which needs are most important and colour coordinated the list with green as the highest priority, amber as the medium priority, and red as the lowest priority. Figure 2 shows the table regarding the prioritisation list.

Requirement Number	Requirement Type
1	FR
2	FR
3	FR
4	FR
5	FR
6	FR
7	FR
8	NFR
9	NFR
10	NFR

Figure 2: Table highlighting the requirements, refer to Appendix D for the document

5.1.3 Functional and Non-functional requirements

After conducting questionnaires with potential users and conducting research on current applications, the author created a list of requirements. These requirements were then put into a Volere template format, which included what the requirement is, a brief description of the demand, a key to identify each Volere template, the individual who suggested the requirement, and how it was tested. A total of 10 were created, 7 functional and 3 non-functional and is found in Appendix D.

Figure 3 is an example of a functional requirement that outlines the system's behaviour. It specifies the application's capability to allow users to submit protest details on a standard account. An account creation and for the user to login is the bare minimum for the task to be carried out. The system will then send the protest details the standard user entered over to the admin which they can view on their account and can then approve the submission.

<table>
<tr><td colspan="2">Requirement ID: 3</td><td>Requirement Type: FR</td></tr>
<tr><td colspan="3">Description: The user should be able to login and submit protest details</td></tr>
<tr><td colspan="3">Rationale: This will allow a standard user to have the ability to submit as much details as they can so it can be approved by the admin</td></tr>
<tr><td colspan="3">Originator: Project manager</td></tr>
<tr><td colspan="3">Fit Criteria: The user will have the ability to log into their account or create one if applicable and navigate to the ‘Tell us about a protest tab’. The user can then fill in as much details as possible in order to submit a protest.</td></tr>
<tr><td colspan="2">Customer Satisfaction: 4</td><td>Customer Dissatisfaction: 4</td></tr>
<tr><td colspan="2">Priority: Essential</td><td>Conflicts: None</td></tr>
<tr><td colspan="2">Supporting Material: None</td><td rowspan="2">Volere

Source: Atlantic Systems Guild</td></tr>
<tr><td colspan="2">History: None</td></tr>
</table>

Figure 3: An example of a functional requirement, refer to Appendix D

Figure 4 is an example of a non-functional requirement, which outlines how the system is performing and is used. It specifies how quickly the programme should react when the admin user adds a protest annotation onto the map. The results should be instant after clicking on the update map button.

Requirement ID: 8	**Requirement Type: Non- Functional**
Description: Details approved by the admin and created on the map view as an annotation should be an instant change once the 'update map' button is clicked.	
Rationale: This will allow the users to see the most immediate changes, so they know whether to attend the upcoming protest or not.	
Originator: Project manager	
Fit Criteria: When an admin updates the details for an upcoming protest, the results should be instant, and a point of interest should be called onto the map for all users.	
Customer Satisfaction: 4	**Customer Dissatisfaction:** 4
Priority: Medium	**Conflicts:** None
Supporting Material: None	Volere Source: Atlantic Systems Guild
History: None	

Figure 4: An example of a non-functional requirement, refer to Appendix D

5.2 Design

5.2.1 User Interface

The project follows a simple design with the quantity of content in the background kept to a minimum to keep the app functional and appealing. FluidUI, an online tool was utilised to create prototypes for the UIs. It was preferable that the application had finished an initial design phase before beginning development in the IDE, so that the design could be replicated into the UI functionalities in XCode. Each UI was created with the colour scheme in mind. Below, Figure 5 shows three wireframes out of the total fourteen frames and the remaining is in Appendix G.

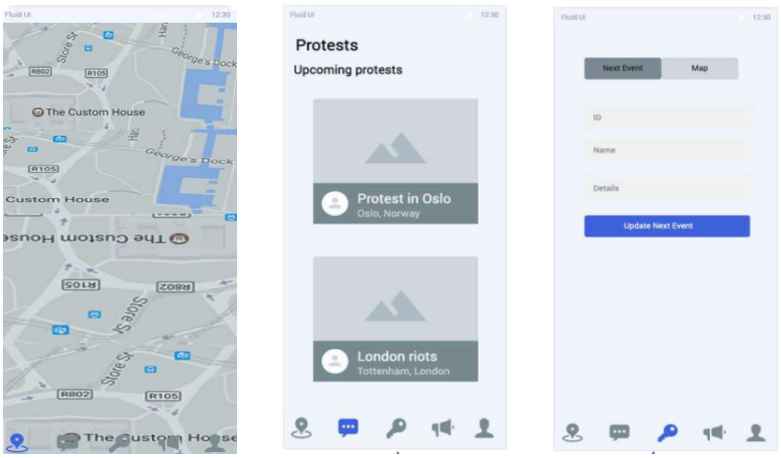


Figure 5: A segment of the wireframes, refer to Appendix G for Wireframes

5.2.2 Navigation Architecture

The use of the Navigation Architecture enabled the author to understand how both the user and admin would browse around the application. It was crucial since as it allowed the author to combine the many tabs produced throughout the UI phase and add any additional functionality that were needed. Refer to Appendix G for the Navigation Architecture design.

5.2.3 Use Case Diagram

As mentioned previously in section 4.1.3, the author utilised Use Case Diagrams (UCD) to show how primary and secondary actors interacted with the use cases in the system. The version of UCD the author produced contains 3 main uses cases, primary actors (User (Customer, Admin)) and secondary actors (FirebaseAuthentication, FirebaseStorage, FirebaseFirestore). Below Figure 6 illustrates the use case diagram used for this project.

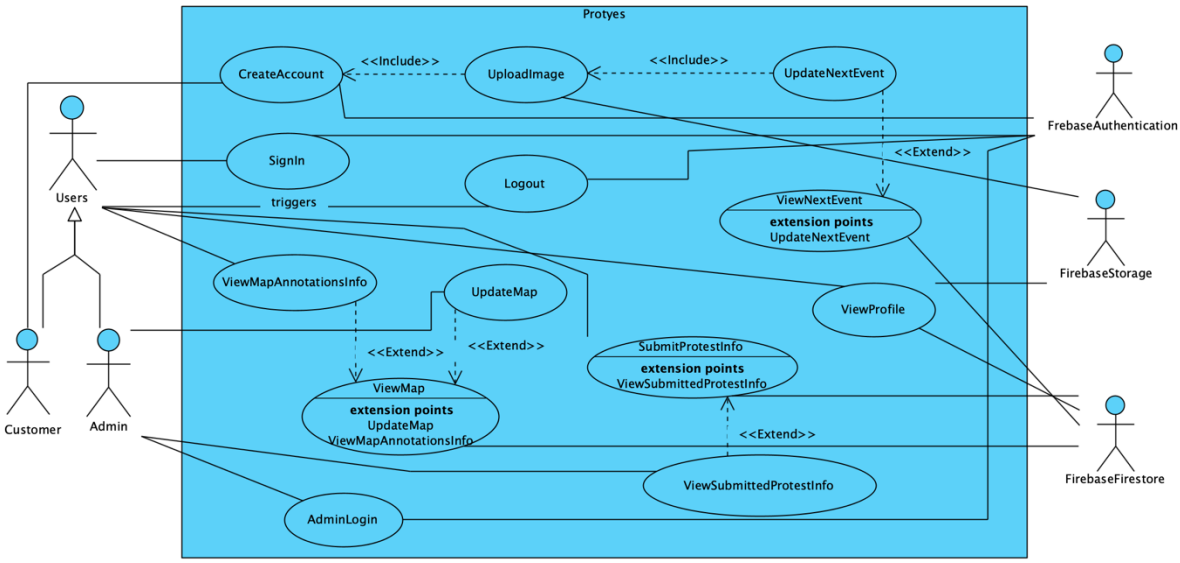


Figure 6: Use case diagram, refer to Appendix G

5.2.4 Use Case Specification

Following the use-case diagram, the author generated a use case specification for the application's major functions, ensuring that the function is built to a high level and that the primary objectives laid out at the outset are met. The Use Case Specification contained information such as the actors, a brief description, pre and post conditions and alternative flows. There were 4 main use case specifications and 4 alternative flows created. Appendix G shows the contents of the specification clearly.

5.2.5 Sequence Diagram

A sequence diagram was utilised to represent the many activities that a user can take as well as the path that would be needed to accomplish these tasks. The login sequence diagram in Figure 7 is an example of this; it displays the route the user must follow when logging in, which involves entering their email and password, as well as how the application interacts with Firebase authentication and at which phases. Once the login information is given to Firebase authentication, there are two possible outcomes, as shown by the alternative box. The first is that the user information is correct, and the user is routed to the home page. If, on the other hand, firebase Authentication does not grant the user access based on the information supplied, an error message is returned.

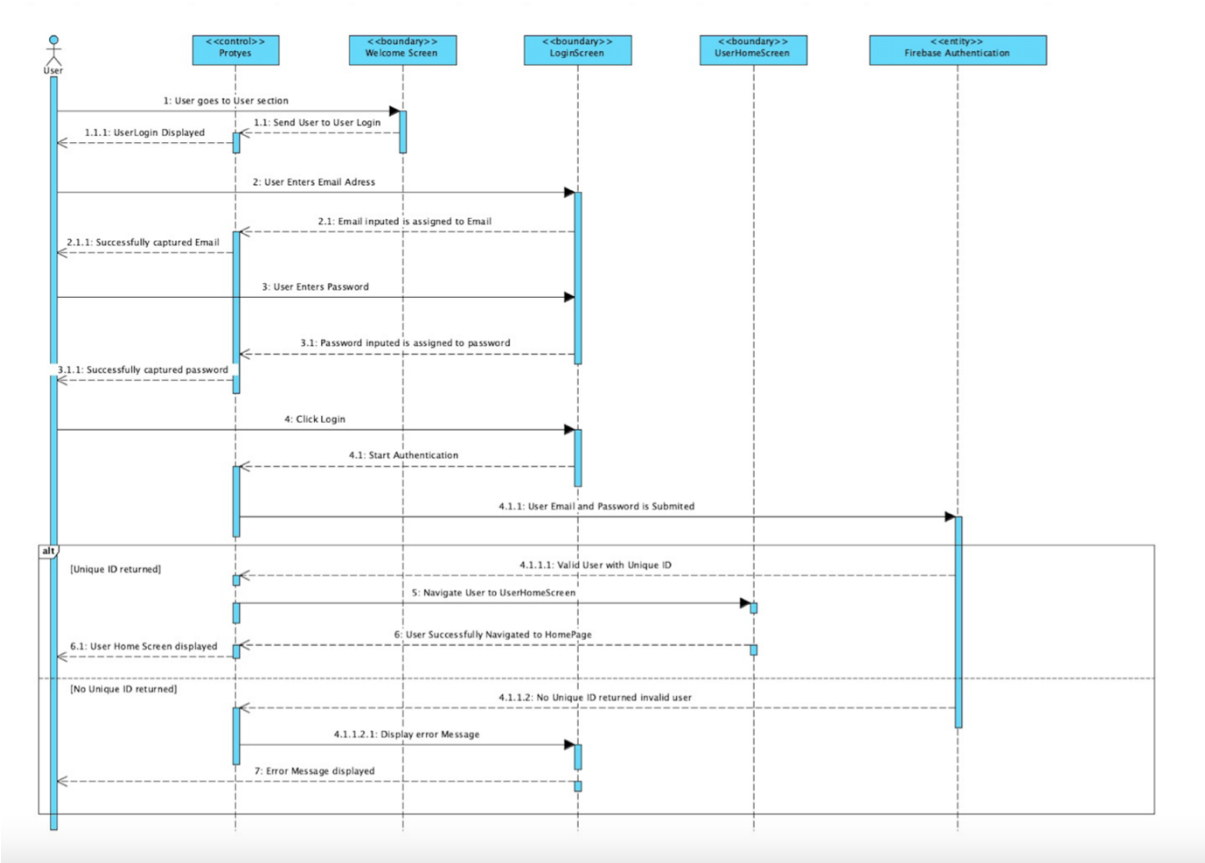


Figure 7: A log in Sequence diagram, refer to Appendix G for remaining

5.2.6 Class Diagram

To have a visual idea of the many classes in the app and how they work together to execute tasks, the author produced a class diagram. To read and write data, authenticate, and store images from Firebase, all class diagrams utilise the service_database package, and inside this package, all classes use the firebaseManager. Appendix G contains this information.

5.2.7 Entity Relationship Diagram

Both Firebase authentication and Firestore are databases that store data. To demonstrate how these many collections and documents function together, the author constructed an Entity Relationship Diagram. The user information, for example, is maintained on Firebase Authentication and allows them access to the Firestore Database after they sign up. When a user is granted access to the Firebase Database, a collection is automatically generated with a document including the user's email address, image, and unique ID. Since only one document is produced per user per collection, this document has a one-to-one relationship, which is demonstrated on the diagram. Figure 8 below shows the results of this diagram.

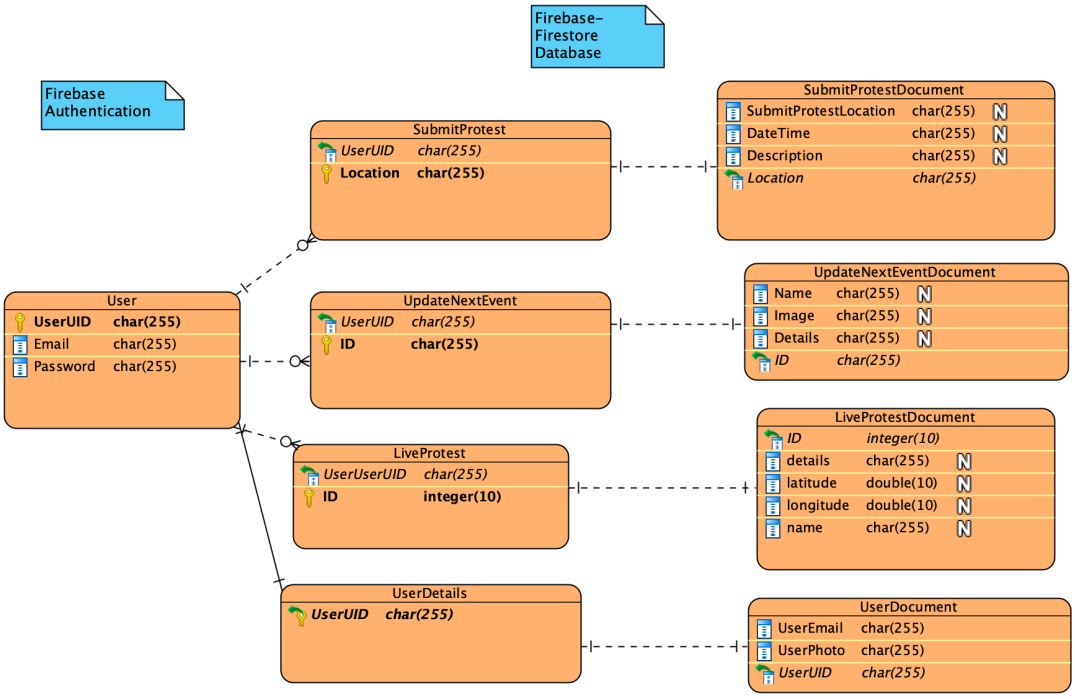


Figure 8: Entity Relation Diagram, refer to Appendix G

5.2.6 Cloud Architecture

The author used Google's Firebase Cloud System, which includes several different services. Authentication, storage, and database are just a few of the services available. As a result, the author created a Cloud Architecture to coordinate the systems' interactions. It clearly indicates how the different services interact with the application. The Figure 9 displays the Cloud Architecture diagram the author assembled to show the functions of the Firebase system.

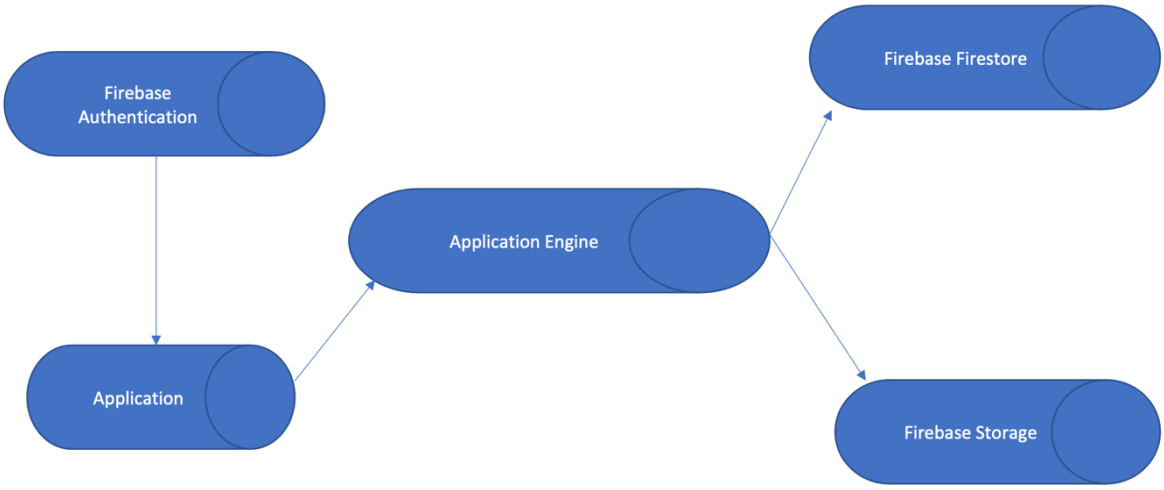


Figure 9: A representation of the Cloud Architecture Diagram

5.3 Implementation

5.3.1 Version Control Software

The work was divided up into increments due to the author's use of the SCRUM technique, and GitHub was used as the platform to handle these changes. The modifications were submitted to GitHub as soon as a work was completed, and the author has previously used this software for other projects and was comfortable with the idea. The project's contents were saved in a private GitHub repository, which allowed each file's change history to be tracked. This link for this will be found in Appendix J.

5.3.2 Sprint 1: Foundations

The first sprint included the connection of Firebase to XCode and creating the graphical user interface (GUI) for the different tabs as mentioned in section 4.3.1. Due to having the Wireframes created at the start of the project, the author was able to replicate this vision on XCode. The initial steps of creating Protyes consisted of using the IDE XCode and creating a new project on that platform. A wizard which prompted for the project name, type and storage location appeared and the author followed through the steps to generate the empty project ready to be developed.

5.3.2.1 Dependencies

To enable third-party API integration, such as Firebase, several dependencies were added to the XCode project. A fragment from the project's dependencies is shown in Figure 10. The Firebase 8.12.1 entry, for example, allows this project to interact with the cloud-based Firebase Realtime database. Another dependency which the author utilised was SDWebImageSwiftUI which could display the URL of the images stored in Firebase Storage. Firebase Firestore, Firebase Authentication,

and Firebase Storage are also included. The author made a document which clearly indicates how to install the dependencies on XCode for beginners and this can be found in Appendix I.

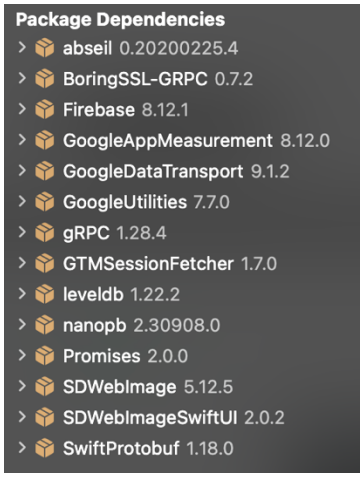


Figure 10: Dependencies added to the XCode project

5.3.2.2 Firebase connection to XCode

When establishing a connection from Google Firebase to XCode, a Bundle Identifier was required by Firebase, and this could be located under the main folder in XCode. Upon entering the Bundle ID, the author proceeded to downloading a PLIST file and incorporated the downloaded file to the XCode project. Once this was done, there was a basic integration from Firebase to XCode. To further facilitate the authentication, the author proceeded to enable 'Email/Password' as Sign-in providers on Firebase as shown in Figure 11. This meant that the author could potentially have the users create accounts and log into those respective accounts.

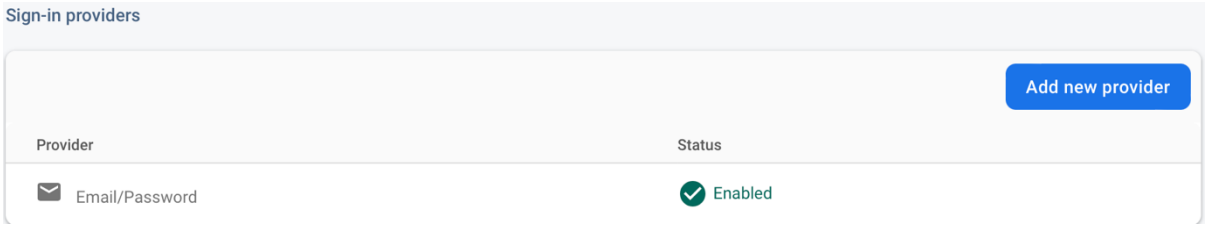


Figure 11: Enabling 'Email/Password' as a Sign-in provider

On XCode, the author included a file called FirebaseManager which essentially laid out the basis for Firebase as it needs to be initialised before it could be used by XCode. Furthermore, Firebase Authentication, Firestore and Storage are also initialised within the Firebase Manager, and these can therefore be accessed using the Firebase Manager class and the same code is not therefore replicated in order to access the Firebase services. This is shown in source code in Appendix I under heading FrebaseManager.

5.3.2.3 Navigation

After completing the successful connection from Firebase to XCode, the author moved onto implementing a functional navigation system. This entailed the basic movement from the tabs that were visually presented on the Wireframes during the design stage. The standard user had 4 different navigation tabs whereas the admin user had a view with 5 navigation tabs. Both had shared access to the Live Map view, the Next Events, the Protest submission view and the profile view. Figure 12 is a screenshot of the user interface for the standard user. Figure 13 is a screenshot of the user interface for the admin user. The extra added feature to the admin was the admin view which enabled the admin to edit and submit details onto the Live Map and Next Events view.

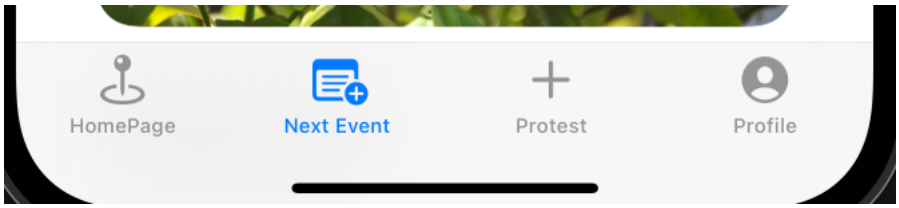


Figure 12: Image representing the standard user navigation

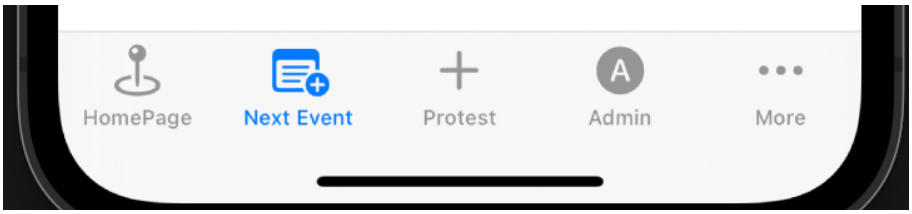


Figure 13: Image representing the admin user navigation

5.3.3 Sprint 2: Map Integration/ Admin Feature

Following the first sprint which covered the foundations of the application, the second sprint intended to create the interactive functions which built on from the navigation system in the first sprint. The second sprint covers the characteristics such as implementing the Map functionalities and admin features.

5.3.3.1 Implement Apple Map

The main purpose of the application was the interaction with the Map feature as users would extract the protest details from here and is the screen that proceeds the login screen. The author utilised the MapKit framework to determine point of interests on the Live Map. In Appendix I, heading HomePageView shows the code which includes the report of importing the framework MapKit into the project.

Following this, the author was then able to present the view of the Apple Map. In the final version of the application, the location of the user is fixed to the coordinates of London, and this has been hardcoded in the development of the project as shown Appendix I, heading HomePageView. The author utilised the website 'Latlong' to find the precise position (Latitude and Longitude Finder, 2022). The current fixed location can be altered to any other location through XCode but in future updates, should the user enable their location while using the app, the location would change according to that. This proposed feature would be useful as the user would have quick and direct view of the upcoming protests in their area without having to zoom in and out to different areas.

5.3.3.2 Implement Live Protest

As a dominant feature of the app, showing the live protest annotations on the map was imperative to the user interaction. To achieve this, the author watched a YouTube video titled 'SwiftUI: Fetching data from Firestore in real-time' which allowed the author to write functions that connected to Firestore and subscribed to any changes that happened. The function 'fetchCities' obtains the data from the database going into the Collections titled 'ProtestLive' and retrieving the information from here. In Appendix I, under the title HomePageView the code prompts the program to return the details entered by the user and display it on the map.

5.3.3.3 Implement Admin Features

An average user would not be able to annotate points of interest on the Live Map view, hence the Admin View on the admin profile is used. A feature solely available to the admin, the author was able to integrate this so the admin can monitor the protests that are submitted and approve them. As this is a prototype, there is one admin password of 'Password', and the admin uses this to log into the profile and approve submissions, effectively displaying them on the map view for all users. This is to imitate the real-life scenario of having the local authority's approval but seen as this is just the first version, one sole admin with can approve and because they are all tests there is no legal permission needed. The Admin view also has the functionality to update the Next Events page which essentially is a blog page that keeps the users up to date with imminent events, as seen in Figure 14, where the image on the left is for the Map update and the one on the right is the Next Event update. In terms of developing code for this feature, the author utilised Firebase official documentation and adapted it to fit the theme { https://firebase.google.com/docs/database/ios/read-and-write}. From this official document, the author customised it for it to be adequate with the vision of how the admin view was to be developed.

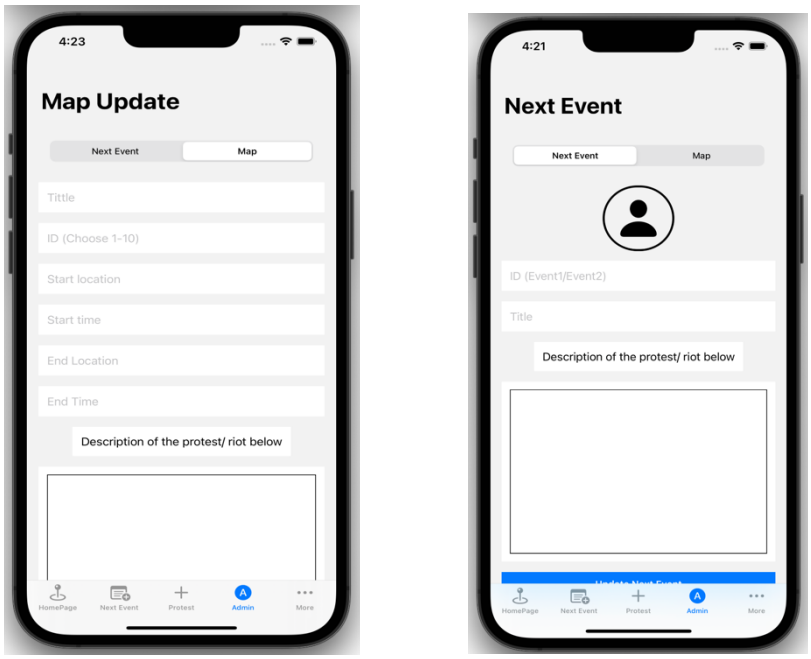


Figure 14: Admin View, left updates map and right updates next events

5.3.3.4 Implement Next Events

Following on from the Admin View feature, upon entering the approved data, the contents written in the 'Next Event' section by the admin and therefore stored in the Firestore database was retrieved and displayed on the Next Events page. This was created for the purpose of displaying 'blog' like information on the application which was readily available for all users to access. The author utilised the YouTube video 'SwiftUI Simple Blog App' which paved the premise of developing the code for Next Events. In Appendix I, under heading NextEvent, the file highlights the code that was written in order to retrieve an image that was fetched from Firebase Storage and display it onto the page adjusting the size and positioning of the image and text. The source code mentions the variable 'details' which refers to the details field that is present on the Admin View page. As mentioned before, the information entered on the admin page is sent to the database then fetched and displayed on the Next Events page.

5.3.4 Sprint 3: User Interaction

After finishing the application's primary components, such as the Live Map and the Admin page, the author focused on creating a method for users to easily register.

5.3.4.1 Login Page

There are different avenues to log into the Protyes application. The first stage, as shown in Appendix I, the heading LoginView, is to create an account which ensures that the information supplied to Firebase Authentication by the email and password controllers is in an acceptable format.

If the user has successfully established their account, a Unique Identifier (UID) is returned; otherwise, an error message is sent back with the problem that happened, such as incorrect format, email already in use, or password does not fulfil the security criteria. The user will be allowed to log in using this information when their UID has been validated using a conditional statement. Furthermore, once the account is created, the user's image is uploaded to Firebase Storage in the users' folder, and a uniform resource link (URL) is generated. Every component such as the email, UID and the image URL link is then stored in the Firestore database.

5.3.4.2 Profile page

The profile page displays important information about the user, such as the user's email address and an image that was chosen during the sign-up process. To show this information, it was taken from Firestore, in the User collection, which is made up of information acquired on users when they signed up, with their unique id serving as the primary key that separates the many user documents. The author utilised the package dependency SDWebImage to show the user profile image, which will obtain the URL of the user profile image kept in Firebase storage and display it on the app. The block code in Appendix I shows how the profile page was created.

5.3.4.3 Protest submissions

Users can provide vital information via the app by heading to the protest submission page, as shown in Figure 15. After entering the information and pressing the submit button, the data is sent to the Firestore database through a key value pair model, where it will be kept and only an admin user will be able to access it using the unique identification produced based on the protest location and date. Once the Admin gets this information, they may conduct more research and utilise other capabilities of the app, such as live map updates and next event updates, to notify users of the protest.

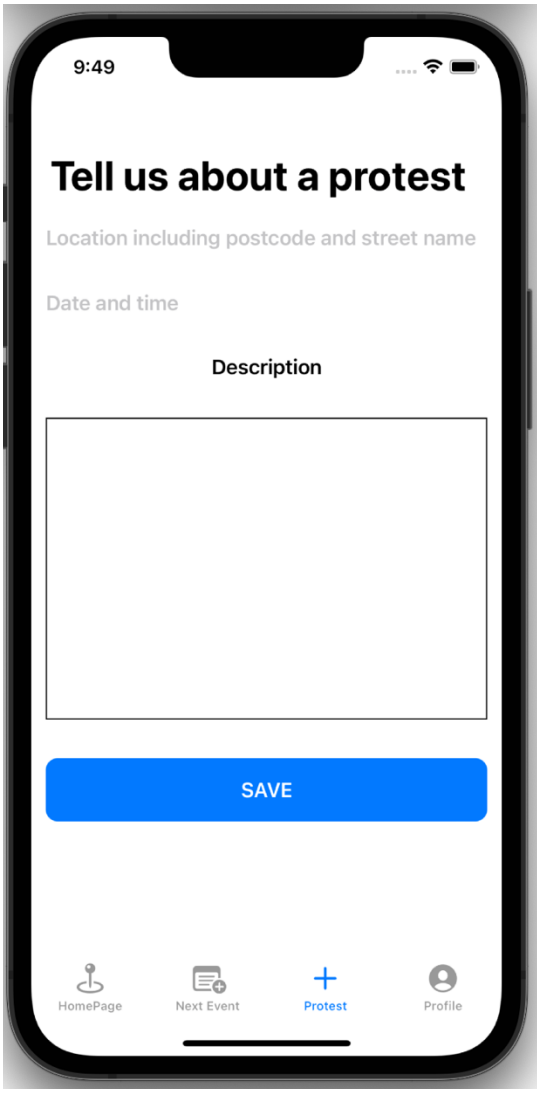


Figure 15: Image showing how to navigate to screen that submits protests

5.3.4.4 Display Protest submission

Once the user has submitted any protest information the admin can see the information displayed on the application through the submitted protest information file which is in their profile. Only admins can see this because admins have a different navigation system compared to standard user and therefore have access to more features. In order to display the information, a query of all the documents in the protest submission collection is made and key information is extracted from these documents such as ‘location’, ‘dateTime’ and ‘description’ this information is then displayed using a list. This is shown in the block code in ProtestInfo which is in Appendix I.

5.3.5 Sprint 4: Finalisation

The author's final sprint focused on project completion; this simply guaranteed that the project's quality was up to standard and that the project was taking shape.

5.3.5.1 Implementing Protest Route

One of the final features to be added to the application was a start and finish route that a protest would take. Unfortunately, the author was unable to fully integrate this feature into the application in time; however, the author was able to make significant progress on this feature, such as importing the latitude and longitude of the start and finish routes into the application and creating a class that can draw and calculate a line based on the two sets of coordinates given for the start and end of the route. When the author attempted to integrate all these features into the application, a problem occurred. After determining that the difficulty stemmed mostly from the map design's architecture and implementation, the author decided that the feature would have to be reassessed, and Figure 16 shows how the feature will appear once implemented.

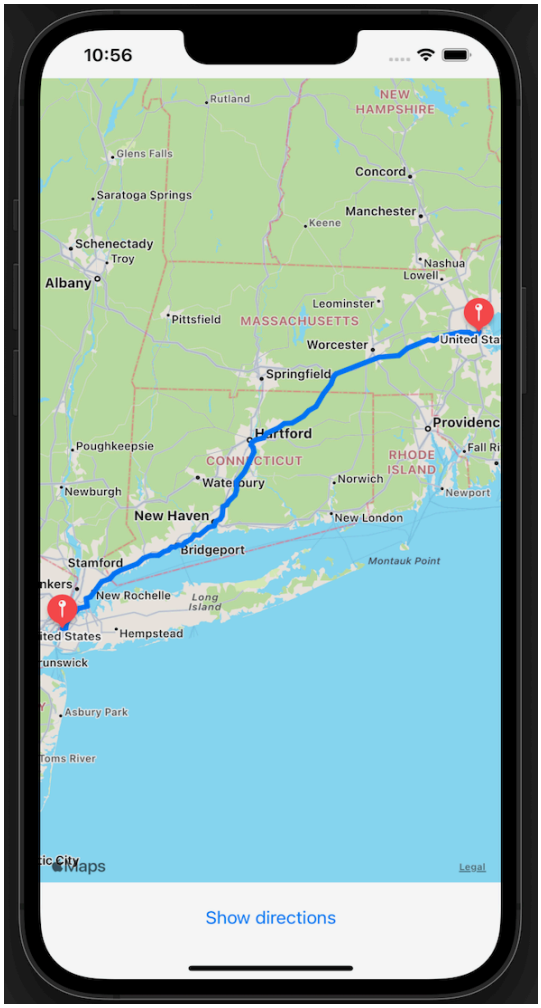


Figure 16: Protest route

5.3.5.2 Fixing any Errors

The process of fixing errors was a task the author conducted throughout the entirety of the project. Due to it being the first time the author utilised Swift there was bound be to mistakes made along the way. To tackle this, the author introduced sprints to allow the author to focus on one section at a time. But even with this framework, whilst coding the sections within them, the author ran into errors. In terms of fixing the errors, the author would route down the avenue of firstly identifying the line in the code where the error occurs, then XCode would prompt the author as to the type of error, next the author could proceed to attempting to eliminate the bug. Version control enabled the author to monitor the changes which essentially meant that the code could be regulated. If the bug was not a simple fix, there were ways in which the author sought to fix the bug like going on Google and Stack Overflow. In the last sprint, the author made sure that any small problems, such as unneeded variables or packages, were cleaned up to improve the application's speed.

5.3.5.3 Improving User Interface

Upon finishing the contents of the application, the author prioritised in making the user interface more appealing. The author followed through with the guidelines of Shneiderman, who claims that characteristics like good usability were critical during the design phase (Gong and Tarasewich, n.d.). As a user's desire to decrease the number of interactions and enhance the speed of engagement grows, the notion of allowing regular users to employ shortcuts becomes increasingly important. The author made sure that the buttons incorporated were simple and that they swiftly navigated the user quickly to the correct place. The navigation features the author included is simple and is presented using icons. The colours blue and white were consistent throughout the application which gives the user a sense of uniformity. The author aimed to improve the text field names to more detailed ones as the first versions deemed to be vague. As well as changing the names, the sizes of the text field were altered to fit the framework for the Next Events page this is shown in Figure 17 and 18 below. Minor changes like these had an effective impact on the overall of the design of Protyes.

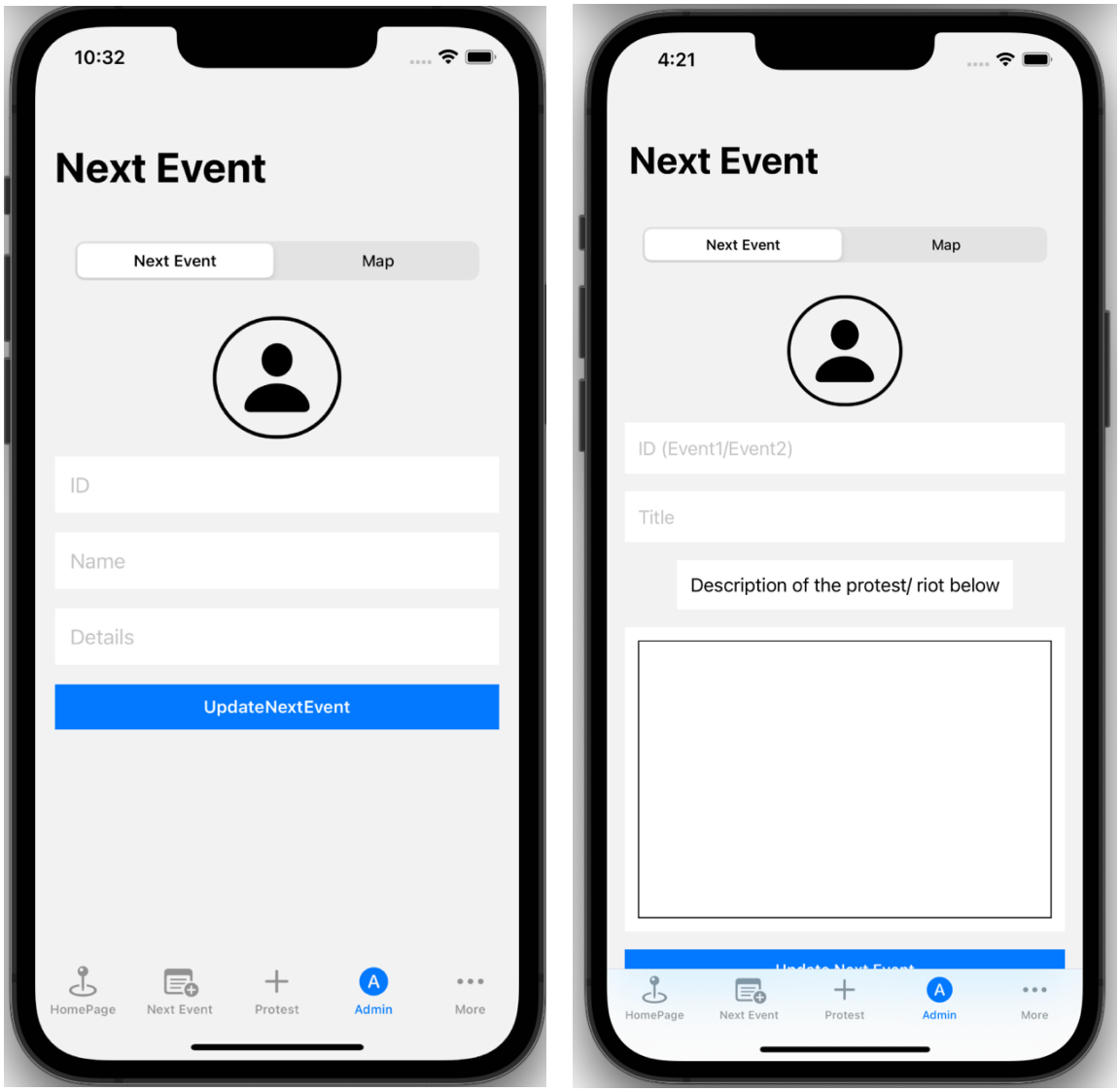


Figure 17 (left) & 18 (right): left shows original text size, right shows modified version

5.4 Evaluation

This section outlines the methods used by the author to keep track of the project's performance and identify the project's relevance and level of achievement, as well as development effectiveness.

5.4.1 Testing

The author did continuous testing and used the four sprints mentioned earlier to test in increments and ensure the project's quality. The testing consisted of ensuring that all the application's features were completely functional and that no defects were detected. The author accomplished this by conducting User Acceptance Test found in Appendix E, Usability Testing and creating a collection of Test Cases. This section explains how the author conducted the tests and what the outcomes were.

5.4.1.1 Usability Testing

The author used Usability Testing to confirm that the product's functions, features, and general purpose are in accordance with what user's desire by watching how people would interact with the project. To do this, the author enlisted the help of eight volunteers. Prior to the testing, the participants were given information sheets and consent forms. After signing the consent forms and double-checking that everything was in order, the author gave the participants instructions on how to set up Firebase and XCode so that the software could run smoothly. Upon successfully installing XCode, the participants were free to explore and familiarise themselves with the application. Following completion, the author distributed a Usability Testing questionnaire to the testers, which allowed them to score the application's design, usability, colours, text size and functionality, as well as answer questions. Appendix E contains all the usability testing questionnaires.

5.4.1.2 Test Cases

The author prepared 9 Test cases to help with the testing stage of the project. These cases outlined how to test the application. The test cases were created with the intention of providing a framework for an action that a tester may perform to confirm a certain feature of a product or application's functionality. The Figure 19 below shows an example of a test case which validates the 'Testing Get Started Button'. All the other Test Cases are in Appendix E.

Objective: Test Live Map Marker button	**Test Number:** 3
Set Up: Ensure the tester (author) is on the Live Map and that they can click on the red marker on the map. The author will then be presented with the information of the protests. 3 different markers will be tested 3 times to ensure consistency and that it swiftly migrates to the protest information screen.	
Expected Results: Correct point is added to the Live Map No time delay when the marker is pressed	
Test: The tester will launch the application and login. Once logged in, the tester is presented with the Live Map page (homepage) and clicks on a marker to present the tester with the information of the protest. 3 buttons will be tested 3 times to ensure consistency and that it swiftly changes over to the protest information screen.	
Test Record: Test successful	
Date: 21 April 2022	**Tester:** Suad Warsame
Result: Passed	

Figure 19: Example Test Case, refer to Appendix E for remaining

5.4.1.3 Test Case Results

Out of all the 9 Test Cases, 1 failed. The test case that failed was Test Number 4 and this was due to the information not successfully appearing when the button was pressed. Upon entering the details to be updated for the Next Event, the tester proceeded to click on the 'Update Next Events' button and the expectation was that they content just submitted would appear on the Next Events page. But this was not the case. There was no update and in fact, the user had to log out and log back into to make these changes apparent. The rest of the test cases were successful, and they are in Appendix E.

Chapter 6: Conclusion

This chapter focuses on the project's overview, as well as a discussion of the general project objectives. The project's advantages and disadvantages are also examined.

6.1 Main Objective

The main objective of the project was to create a platform where users could track protests in their area as well as others reported around the world. The map was to be implemented using the Apple MapKit and include pointers which alerted users of the protests. The objective has been met and Protyes successfully includes this functionality. Users can interact with the Live Map which is on the homepage by successfully submit upcoming protests which will then be sent and verified by the admin.

6.1.2 Sub-objectives

The sub-objectives included components like the login page, profile page, next events page, and admin page.

6.1.2.1 Sub-Objective: Login Page

The Login page was created to give a platform for the user to register and sign in. It was also created so that the administrator could access their account. Both objectives were achieved, and the user was able to log into the software with ease.

6.1.2.2 Sub-Objective: Profile Page

The user's profile page was created so that they could view components that they had personalised, such as the image they picked as their profile picture and their email address. The possibility to sign out was offered here, and the author fulfilled the criteria successfully.

6.1.2.3 Sub-Objective: Next Events Page

The future events page was established to show the most updated news on planned protests. The author has accomplished this goal, and the user may now explore the blog-like information available in this tab.

6.1.2.4 Sub-Objective: Admin Page

The admin page offered the admin the option to customise the information on the upcoming events page as well as add the appropriate point on the map that showed the planned protests. This goal has been accomplished.

6.2 Knowledge gained

Throughout the entirety of the project, the author was exposed to content that they never utilised before. Software such as XCode was something familiar but the language Swift was not utilised before. Swift was self-taught from visiting online documents and watching YouTube videos to provide an understanding on the basics of the language. The code was then adapted and altered to fit the criteria and objectives the author set out. SCRUM is an agile technique that allows users to work on parts of work in small increments. The author learned that using this was crucial since it allowed the author to establish adequate plans and designs prior to the creation of the project, as well as revise the work and make any necessary modifications. This had an overall improvement on the authors time management thus provided the way to successfully track progress.

6.3 Future projects

Due to this being the first version of Protyes, in future releases, there will be many advanced modifications and additions to the application.

6.3.1 Protest route

A prime feature that will be added to the next release is the Protest Route that has been mentioned in section 5.3.5.1. This would replace the current feature of single points on the map and would overall provide a seamless tracking system. It would visually be more useful as it clearly shows the exact route of the protest rather than having the current designated point which once clicked, provides detail on the protest in text format. This feature was too complex and time consuming to add in this version but is a plan proposed for the next release.

6.3.2 Automatic Coordinates

Another feature of making the location field fill in automatically in terms of the latitude and longitude when adding a protest on the Live Map, is an implementation for the next updates. The coordinates containing the latitude and longitude should populate automatically instead of going on third party websites and sourcing this information. This is very beneficial and from research of the questionnaires, the author gathered that this was essentially in making the usability easier.

6.3.3 Calendar

The notion of a calendar has also been proposed to be included in future versions of the Protyes application. The ability to display all the protests on a calendar that the user may personalise is a feature for future iteration. It was suggested that a calendar function be included that allowed users to organise

and browse events, as well as decide whether to attend the scheduled protest. This simply allows the user to keep track of and stay ahead of crucial dates.

6.4 Conclusions drawn from the project

Before starting Protyes, the author noticed a vacuum in the market: there was no way to follow upcoming protests other than through social media or word of mouth. The author saw that there was a problem that needed to be solved for the world to benefit. This was the beginning of Protyes' existence. Protyes differs from software like Citizen in that it contains an interactive map that users may use to learn about planned protests not just nationally but globally. Citizen, on the other hand, provides a platform that feeds information to people from the perspective of the police, emphasising key information such as riots and protests when they occur rather than as they are planned. Essentially, it is a police walkie-talkie.

In order to create a successful system, the author developed the capacity to operate on a set timetable and fulfil deadlines. The author was able to enhance their time management as a result of this. Furthermore, studying new content helped the author to develop their skills and get insight into internet resources that they could alter and tailor to the project's development, which increased the author's creativity. Not only that, but because this was the author's first time producing a report of this type and magnitude, their writing abilities improved dramatically during the report.

Finally, this report produced a workable prototype for a protest tracking system. The system allowed regular users to enter protest information, that was then reflected on the Live Map by the admin, which was then accessible to all users. All the successful testing, as well as the change of the code to satisfy the objectives stated at the start of the report, resulted in the completion of Protyes.

Reference

Aggarwal, N., 2022. *Introduction to Xcode—iOS Development.*. [online] Medium. Available at: <https://medium.com/adg-vit/introduction-to-xcode-ios-development-6e262dd4e5d8> [Accessed 19 February 2022].

Apple Developer. 2022. *Apple Developer*. [online] Available at: <https://developer.apple.com/> [Accessed 19 February 2022].

Citizen.com. 2022. *Citizen: Keeping you safe & informed*. [online] Available at: <https://citizen.com/about> [Accessed 19 February 2022].

Docs.swift.org. 2022. *About Swift*. [online] Available at: <https://docs.swift.org/swift-book/> [Accessed 20 February 2022].

En.wikipedia.org. 2022. *Apple Developer - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Apple_Developer> [Accessed 19 February 2022].

Firebase. 2022. *Firebase Realtime Database | Store and sync data in real time*. [online] Available at: <https://firebase.google.com/products/realtime-database> [Accessed 5 April 2022].

Firebase. 2022. *Get started with Cloud Storage on Apple platforms | Firebase Documentation*. [online] Available at: <https://firebase.google.com/docs/storage/ios/start> [Accessed 5 April 2022].

Firebase. 2022. *Get Started with Firebase Authentication on Apple Platforms | Firebase Documentation*. [online] Available at: <https://firebase.google.com/docs/auth/ios/start> [Accessed 5 April 2022].

Fueled. 2022. *Meet Countable: The Non-Partisan Political App You Need to Download*. [online] Available at: <https://fueled.com/blog/political-news-app/> [Accessed 21 February 2022].

Gong, J. and Tarasewich, P., n.d. *GUIDELINES FOR HANDHELD MOBILE DEVICE INTERFACE DESIGN*. Boston [Accessed 15 April 2022].

Harding, A., 2022. *South Africa riots: The inside story of Durban's week of anarchy*. [online] BBC News. Available at: <https://www.bbc.co.uk/news/world-africa-57996373> [Accessed 3 February 2022].

Highsmith, J., 2004. *Agile project management*. Upper Saddle River: Addison-Wesley, p.chapter 3. [Accessed 15 April 2022].

Ico.org.uk. 2022. *Guide to the UK General Data Protection Regulation (UK GDPR)*. [online] Available at: <https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/> [Accessed 4 February 2022].

Integrating MapKit with SwiftUI – Bucket List SwiftUI Tutorial 4/12. 2021. [video] Directed by P. Hudson. YouTube [Accessed 5 April 2022].

Petersen, K., 2010. An Empirical Study of Lead-Times in Incremental and Agile Software Development. In: *An Empirical Study of Lead-Times in Incremental and Agile Software Development*. [online] Ronneby, Sweden: Springer, Berlin, Heidelberg, p.1. Available at: <https://link.springer.com/chapter/10.1007/978-3-642-14347-2_30> [Accessed 3 April 2022].

Robertson, J. and Robertson, S., 2012. *Volere Requirements Specification Template Edition* 16., pp. 5 [online] Cs.uic.edu. Available at: <https://www.cs.uic.edu/~i440/VolereMaterials/templateArchive16/c%20Volere%20template16.pdf > [Accessed 13 April 2022].

Rosencrance, L., 2022. *What is Google Firebase? - Definition from WhatIs.com*. [online] SearchMobileComputing. Available at: <https://www.techtarget.com/searchmobilecomputing/definition/Google-Firebase> [Accessed 19 February 2022].

Schwaber, K. and Sutherland, J., 2016. *The Definitive Guide to Scrum: The Rules of the Game*. [online] Scrumguides.org. Available at: <https://scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf#zoom=100> [Accessed 3 April 2022].

Simmons, M., 2022. *The 100-Hour Rule: Forgotten Study Shows How You Can Become World-Class In 100 Hours*. [online] Medium. Available at: <https://medium.com/accelerated-intelligence/the-100-hour-rule-forgotten-study-shows-how-you-can-become-world-class-in-100-hours-ae2f94cc2fb0> [Accessed 4 February 2022].

Stack Overflow. 2022. *Stack Overflow - Where Developers Learn, Share, & Build Careers*. [online] Available at: <https://stackoverflow.com/> [Accessed 19 February 2022].

SwiftUI Firebase Chat 07: Sign Out of Firebase. 2021. [video] Youtube.

SwiftUI Simple Blog App. 2021. [video] YouTube.

SwiftUI with Firebase - Fetch and Save Data. 2021. [video] YouTube.

Volere Requirements. 2022. *Volere Requirements Specification Template – Volere Requirements*. [online] Available at: <https://www.volere.org/templates/volere-requirements-specification-template/> [Accessed 15 February 2022].

Youtube.com. 2020. *SwiftUI Tutorial - MapKit, Route, and Directions*. [online] Available at: <https://www.youtube.com/watch?v=H6pmm62axCg&list=LL&index=26&t=667s> [Accessed 14 April 2022].

Glossary

Abbreviation	Non-abbreviation
API	Application Programming Interface
FR	Functional Requirement
IDE	Integrated Development Environment
IOS	iPhone Operating System
NFR	Non-Functional Requirement
PDD	Project Definition Document
UAT	User Acceptance Test
UCD	Use Case Diagram
UI	User Interface
UID	Unique Identifier
URL	Unique Resource Link

Project Definition Document

Protyes

A mobile application that manages the Protests in your area

Author: Suad Warsame

Suad.warsame@city.ac.uk

Consultant: Giacomo Tarroni

City, University of London

BSc Computer Science

Academic year: 2021-2022

CITY UNIVERSITY LONDON

Contents

Project Proposal:

Problem to be solved:

The focus of this project is to develop an interactive platform that enables the ability for users to track and be updated with any demonstrations, protests or riots that are happening in their area, as well as interacting globally. The application is simple, it is essentially a global protest tracker whereby the organiser of the protest can create an account within the platform and then make an event for their planned demonstration or protest, provided they lawfully have written to the authority to proceed. In terms of riots, it will be slightly different, like Waze in terms of having the feature to pre-warn fellow drivers of accident or police ahead, the users of this application will have the feature to do this, as riots are more unpredictable. A recent example is the South Africa Riots in July 2021, whereby more than 300 deaths were recorded (Harding, 2022). At the time of the riots, communities would get together to prevent looting and further political unrest, had an app that clearly indicated where the looting and riots were taking place, it would be easier and quicker to target. This application aims to allow registered users to give updates which will then be verified by fellow users to confirm its legitimacy. The application will be designed to solely focus on the events of demonstrations and protests happening, taking into consideration accessibility, confidentiality, and availability of information. Demonstrations and protests are prevalent and could happen at any time. Therefore, the planned demonstrations and protests could be updated on the application, meaning anyone looking to join these or even avoid traffic at a certain area would benefit. Usually, people would search on social media networks such as Facebook and Twitter, but the idea of this project is to integrate this information into an interactive map with a live calendar event.

Project Objectives:

The main objective of the application is to provide a service that enables the users to actively add upcoming events such as protests and demonstrations. This project shall have an interactive map which uses Google Map API and integrates it with a Google Calendar API, clearly explaining the events in the chosen area. The Google APIs would be used to connect and fetch content for both the map and calendar interfaces utilising the open-source components to mirror these on the website. It will be global, so information regarding different countries would be available. Research and background information on the topic will be conducted to understand the topic in detail. The objectives that will be followed throughout this project are the following:

Objective 1: Login screen:

- Let user chose between the option of creating a new account or log back into registered account
- Enable user to fill in form to register account if previous account is not attained

Objective 2: Home page

- Incorporate a friendly user interface using Google Maps API to implement this Objective 3:
- Provide a calendar feature using Google Calendar API

The following tools are going to utilised to achieve my objectives:

- Utilising the language Swift: It is a language developed by Apple and this would be applied to the frontend and backend coding
- Applying Google API: This is the main feature of the application, where the JavaScript allows the customisation of the map and calendar with personal content and imagery to display on the proposed mobile devices
- Coding on XCode: Throughout the project, the platform XCode is used to code using the language swift. The setup will require the API Key provided by the Google Cloud Account and there onwards swift is utilised
- Use of Cocoa Pods: It will manage library dependencies for the XCode projects, where the projects are specified in a single text file called PodFile, which will be created
- Documenting on FireStore and FireBase: Any information saved is going to use FireBase and FireStore, which provides storing, tracking, and reporting tools, showing it live. It is a database that stores on Google Cloud
- Creating a Wireframe: This will be useful to create the design of the app. MockFlow will be used to sketch the user interface.

Project Beneficiaries:

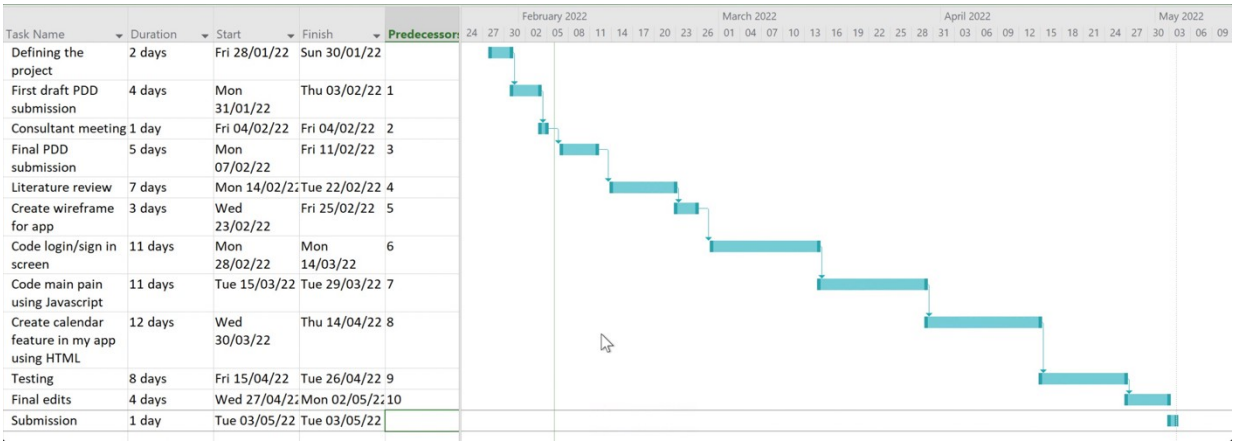
The project will be beneficial in reducing the collateral damage riots leave as users will be more aware of the riots in their area. It will allow users to have a live update of protests they can join which promotes freedom of speech. It will greatly benefit the local authorities as they will have the ability to

utilise their resources effectively for there to be an equal balance in their priorities. It allows the event organisers to plan their events in a professional manner, setting the time, date, and place for their occasion. It will allow the users to actively save the planned dates onto their mobile calendar. Academic students and researchers that have interests in field of political movements can also learn from this application.

Work Plan:

Activities and outputs	Resources	Start date	End Date
Choose project	- Past examples on Moodle - previous knowledge/ ideas	28/01/2022	30/01/2022
Submit first draft of PDD	- Project Handbook	02/02/2022	06/02/2022
Meet with consultant to discuss the Project Definition Document	- Consultant meeting - Project Handbook	08/02/2022	08/02/2022
Submit final PDD	- Project Handbook - Feedback from consultant	08/02/2022	13/02/2022
Literature review	- Online research - Scopus - IEEE Xplore digital library for advance searches	13/02/2022	19/02/2022
Create a sketch of my app (wireframe)	- IN2013 Object oriented analysis and design	20/02/2022	22/02/2022
Start coding the login/sign in screen	- Online resources	23/02/2022	09/03/2022
Using JavaScript, HTML and CSS, begin coding the main page with the map and options to go to other tabs	- Online resources	12/03/2022	25/03/2022
Use HTML and JavaScript to a create my calendar feature in my app	- Online resources	26/03/2022	10/04/2022
Testing (and fixing possible errors)	-	11/04/2022	20/04/2022
Final edits	-	20/04/2022	25/04/2022
Submit	-	04/05/2022	04/05/2022

Image 1



Task Name	Duration	Start	Finish	Predecessors
Defining the project	2 days	Fri 28/01/22	Sun 30/01/22	
First draft PDD submission	4 days	Mon 31/01/22	Thu 03/02/22	1
Consultant meeting	1 day	Fri 04/02/22	Fri 04/02/22	2
Final PDD submission	5 days	Mon 07/02/22	Fri 11/02/22	3
Literature review	7 days	Mon 14/02/22	Tue 22/02/22	4
Create wireframe for app	3 days	Wed 23/02/22	Fri 25/02/22	5
Code login/sign in screen	11 days	Mon 28/02/22	Mon 14/03/22	6
Code main pain using Javascript	11 days	Tue 15/03/22	Tue 29/03/22	7
Create calendar feature in my app using HTML	12 days	Wed 30/03/22	Thu 14/04/22	8
Testing	8 days	Fri 15/04/22	Tue 26/04/22	9
Final edits	4 days	Wed 27/04/22	Mon 02/05/22	10
Submission	1 day	Tue 03/05/22	Tue 03/05/22	

My work ensures that all the objectives are met. Image 1 shows the Gantt Chart which outlines the project objectives. Additional time has been added for mitigating circumstances per task.

Risks to your project:

RISK	CHANCES OF IT HAPPENING 1(LOW) – 5(HIGH)	HOW TO DECREASE THE LIKELIHOOD OF THE RISK HAPPENING	HOW DAMAGE WILL BE LIMITED IF THEY DO HAPPEN
Programming/coding – for this project I may be learning a new software development language. This may result in me not fully understanding the language resulting in errors.	4	By ensuring I do over 100 hours of learning I can decrease the chance of error (Simmons, 2022).	If errors occur due to this risk, I will implement more learning time into the chosen software development language to be able to rectify this.
Loss of work –the workload of developing an app may cases work to be lost or go missing.	2	All work will be saved on an external hard drive to ensure that there is a	If any work is loss during the development process, my timetable will be
		copy in case the work is lost.	re-arranged to make up for the time spent on the lost work.

Time management – by not organising my time efficiently, I may find myself running out of time.	3	I will use diaries and calendars to ensure that I can effectively use my hours. I will also use phone notification reminders and alarms to ensure all tasks are completed.	If time begins to run out before I can complete my application, I will defer all plans during that period to ensure I have my application completed.
General Data Protection Regulations (GDPR) – breach of personal data is a major risk. Need to ensure that all information on app users' identity is kept hidden as to their liking (Guide to the UK General Data Protection Regulation (UK GDPR), 2022).	2	I will be making use of available security features to ensure that all app users' information can be controlled as per what the app user wants	If this breach was to occur, a crisis management plan will commence to ensure that this issue to rectified and the app user's information is kept safe.

Ethics:

Research Ethics Review Form: BSc, MSc and MA Projects Computer

Science Research Ethics Committee (CSREC)

http://www.city.ac.uk/department-computer-science/research-ethics

Undergraduate and postgraduate students undertaking their final project in the Department of Computer Science are required to consider the ethics of their project work and to ensure that it complies with research ethics guidelines. In some cases, a project will need approval from an ethics

committee before it can proceed. Usually, but not always, this will be because the student is involving other people ("participants") in the project.

In order to ensure that appropriate consideration is given to ethical issues, all students must complete this form and attach it to their project proposal document. There are two parts:

PART A: Ethics Checklist. All students must complete this part.

The checklist identifies whether the project requires ethical approval and, if so, where to apply for approval.

PART B: Ethics Proportionate Review Form. Students who have answered "no" to all questions in A1, A2 and A3 and "yes" to question 4 in A4 in the ethics checklist must complete this part. The project supervisor has delegated authority to provide approval in such cases that are considered to involve MINIMAL risk. The approval may be ***provisional*** – *identifying the planned research as* likely to involve MINIMAL RISK. In such cases you must additionally seek ***full approval*** from the supervisor as the project progresses and details are established. ***Full approval*** must be acquired in writing, before beginning the planned research.

A.1 If you answer YES to any of the questions in this block, you must apply to an appropriate external ethics committee for approval and log this approval as an External Application through Research Ethics Online - https://ethics.city.ac.uk/		*Delete as appropriate*
1.1	Does your research require approval from the National Research Ethics Service (NRES)? *e.g. because you are recruiting current NHS patients or staff?* *If you are unsure try - https://www.hra.nhs.uk/approvals-amendments/what-approvalsdo-i-need/*	~~YES~~ / NO

1.2	Will you recruit participants who fall under the auspices of the Mental Capacity Act? *Such research needs to be approved by an external ethics committee such as NRES or the Social Care Research Ethics Committee - http://www.scie.org.uk/research/ethicscommittee/*	~~YES~~ / NO
1.3	Will you recruit any participants who are currently under the auspices of the Criminal Justice System, for example, but not limited to, people on remand, prisoners and those on probation? *Such research needs to be authorised by the ethics approval system of the National Offender Management Service.*	~~YES~~ / NO

A.2 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee, you must apply for approval from the Senate Research Ethics Committee (SREC) through Research Ethics Online - https://ethics.city.ac.uk/		*Delete as appropriate*
2.1	Does your research involve participants who are unable to give informed consent? *For example, but not limited to, people who may have a degree of learning disability or mental health problem, that means they are unable to make an informed decision on their own behalf.*	~~YES~~ / NO
2.2	Is there a risk that your research might lead to disclosures from participants concerning their involvement in illegal activities?	~~YES~~ / NO
2.3	Is there a risk that obscene and or illegal material may need to be accessed for your research study (including online content and other material)?	~~YES~~ / NO
2.4	Does your project involve participants disclosing information about special category or sensitive subjects? *For example, but not limited to: racial or ethnic origin; political opinions; religious beliefs; trade union membership; physical or mental health; sexual life; criminal offences and proceedings*	~~YES~~ / NO
2.5	Does your research involve you travelling to another country outside of the UK, where the Foreign & Commonwealth Office has issued a travel warning that affects the area in which you will study? *Please check the latest guidance from the FCO - http://www.fco.gov.uk/en/*	~~YES~~ / NO
2.6	Does your research involve invasive or intrusive procedures? *These may include, but are not limited to, electrical stimulation, heat, cold or bruising.*	~~YES~~ / NO
2.7	Does your research involve animals?	~~YES~~ / NO
2.8	Does your research involve the administration of drugs, placebos or other substances to study participants?	~~YES~~ / NO

A.3 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee or the SREC, you must apply for approval from the Computer Science Research Ethics Committee (CSREC) through Research Ethics Online - https://ethics.city.ac.uk/ **Depending on the level of risk associated with your application, it may be referred to the Senate Research Ethics Committee.**	*Delete as appropriate*

3.1	Does your research involve participants who are under the age of 18?	~~YES~~ / NO
3.2	Does your research involve adults who are vulnerable because of their social, psychological or medical circumstances (vulnerable adults)? *This includes adults with cognitive and / or learning disabilities, adults with physical disabilities and older people.*	~~YES~~ / NO
3.3	Are participants recruited because they are staff or students of City, University of London? *For example, students studying on a particular course or module.* *If yes, then approval is also required from the Head of Department or Programme Director.*	~~YES~~ / NO
3.4	Does your research involve intentional deception of participants?	~~YES~~ / NO
3.5	Does your research involve participants taking part without their informed consent?	~~YES~~ / NO
3.5	Is the risk posed to participants greater than that in normal working life?	~~YES~~ / NO
3.7	Is the risk posed to you, the researcher(s), greater than that in normal working life?	~~YES~~ / NO
A.4 If you answer YES to the following question and your answers to all other questions in sections A1, A2 and A3 are NO, then your project is deemed to be of MINIMAL RISK. **If this is the case, then you can apply for approval through your supervisor under PROPORTIONATE REVIEW. You do so by completing PART B of this form.** **If you have answered NO to all questions on this form, then your project does not require ethical approval. You should submit and retain this form as evidence of this.**		*Delete as appropriate*
4	Does your project involve human participants or their identifiable personal data? *For example, as interviewees, respondents to a survey or participants in testing.*	YES / ~~NO~~

PART B: Ethics Proportionate Review Form

If you answered YES to question 4 and NO to all other questions in sections A1, A2 and A3 in PART A of this form, then you may use PART B of this form to submit an application for a proportionate ethics review of your project. Your project supervisor has delegated authority to review and approve this application under proportionate review. You must receive final approval from your supervisor in writing before beginning the planned research.
However, if you cannot provide all the required attachments (see B.3) with your project proposal (e.g. because you have not yet written the consent forms, interview schedules etc), the approval from your supervisor will be ***provisional***. You **must** submit the missing items to your supervisor for approval prior to commencing these parts of your project. Once again, you must receive written confirmation from your supervisor that any provisional approval has been superseded by with ***full approval*** of the planned activity as detailed in the full documents. **Failure to follow this procedure and demonstrate that final approval has been achieved may result in you failing the project module.**

Your supervisor may ask you to submit a full ethics application through Research Ethics Online, for instance if they are unable to approve your application, if the level of risks associated with your project change, or if you need an approval letter from the CSREC for an external organisation.

B.1 The following questions must be answered fully. All grey instructions must be removed.		*Delete as appropriate*
1.1.	Will you ensure that participants taking part in your project are fully informed about the purpose of the research?	**YES / ~~NO~~**
1.2	Will you ensure that participants taking part in your project are fully informed about the procedures affecting them or affecting any information collected about them, including information about how the data will be used, to whom it will be disclosed, and how long it will be kept?	**YES / ~~NO~~**
1.3	When people agree to participate in your project, will it be made clear to them that they may withdraw (i.e. not participate) at any time without any penalty?	**YES / ~~NO~~**
1.4	Will consent be obtained from the participants in your project? Consent from participants will be necessary if you plan to involve them in your project or if you plan to use identifiable personal data from existing records. "Identifiable personal data" means data relating to a living person who might be	**YES / ~~NO~~**

	identifiable if the record includes their name, username, student id, DNA, fingerprint, address, etc. *If YES, you must attach drafts of the participant information sheet(s) and consent form(s) that you will use in section B.3 or, in the case of an existing dataset, provide details of how consent has been obtained.* *You must also retain the completed forms for subsequent inspection. Failure to provide the completed consent request forms will result in withdrawal of any earlier ethical approval of your project.*	
1.5	Have you made arrangements to ensure that material and/or private information obtained from or about the participating individuals will remain confidential?	**YES / ~~NO~~**

B.2 If the answer to the following question (B2) is YES, you must provide details		*Delete as appropriate*
2	Will the research be conducted in the participant's home or other non-University location? *If **YES**, you must provide details of how your safety will be ensured.*	**~~YES~~ / NO**

B.3 Attachments **ALL of the following documents MUST be provided to supervisors if applicable.** **All must be considered prior to final approval by supervisors.** **A written record of final approval must be provided and retained.**	***YES***	***NO***	***Not Applicable***
Details on how safety will be assured in any nonUniversity location, including risk assessment if required (see B2)			**X**
Details of arrangements to ensure that material and/or private information obtained from or about the participating individuals will remain confidential (see B1.5) *Any personal data must be acquired, stored and made accessible in ways that are GDPR compliant.*			**X**
Full protocol for any workshops or interviews**			**X**
Participant information sheet(s)**	**X**		
Consent form(s)**	**X**		

Questionnaire(s)** *sharing a Qualtrics survey with your supervisor is recommended.*	**X**		
Topic guide(s) for interviews and focus groups**	**X**		
Permission from external organisations or Head of Department** *e.g. for recruitment of participants*			**X**

***If these items are not available at the time of submitting your project proposal, then **provisional approval** can still be given, under the condition that you must submit the final versions of all items to your supervisor for approval at a later date. **All** such items **must** be seen and approved by your supervisor before the activity for which they are needed begins. Written evidence of **final approval** of your planned activity must be acquired from your supervisor before you commence.*

Changes

If your plans change and any aspects of your research that are documented in the approval process change as a consequence, then any approval acquired is invalid. If issues addressed in Part A (the checklist) are affected, then you must complete the approval process again and establish the kind of approval that is required. If issues addressed in Part B are affected, then you must forward updated documentation to your supervisor and have received written confirmation of approval of the revised activity before proceeding.

Templates for Consent and Information

You must use the templates provided by the University as the basis for your participant information sheets and consent forms. You **must** adapt them according to the needs of your project before you submit them for consideration.

Participant Information Sheets, Consent Forms and Protocols must be consistent. Please ensure that this is the case prior to seeking approval. Failure to do so will slow down the approval process.

We strongly recommend using Qualtrics to produce digital information sheets and consent forms.

Further Information http://www.city.ac.uk/department-computer-science/research-ethics

https://www.city.ac.uk/research/ethics/how-to-apply/participant-recruitment

https://www.city.ac.uk/research/ethics

Participant Information Sheet (Protyes)

REC reference number, date and version of information sheet

REC reference number: -

Date: 13th of February 2022

Version 1.0

Title of study

Protyes – A mobile application that manages the protests in your area

Name of principal investigator/researcher Suad Warsame

Invitation paragraph

We would like to invite you to take part in a research study. Before you decide whether you would like to take part it is important that you understand why the research is being done and what it would involve for you. Please take time to read the following information carefully and discuss it with others if you wish. Ask us if there is anything that is not clear or if you would like more information. You will be given a copy of this information sheet to keep.

What is the purpose of the study?

The purpose of this study is to manage the upcoming Protests in your area and have the ability to check previous dates. It will provide the user with the ability to add these dates onto their calendar making it much easier to track.

Why have I been invited to take part?

The prime reason for this is to conduct research and feedback on the prototype during testing. This is purely to gather thoughts on how the application is functioning and participating will not have any effect on your assessments, marks, or future studies.

Do I have to take part?

Participation in the project is voluntary, and you can choose not to participate in part or all of the project. You can withdraw at any stage of the project without being penalised or disadvantaged in any way. It is up to you to decide whether or not to take part. If you do

decide to take part, you will be asked to sign a consent form. If you decide to take part, you are still free to withdraw at any time and without giving a reason.

If appropriate, include that once the data has been anonymised/published participants will no longer be able to withdraw their data. If the study is only collecting anonymous information, for instance a paper based or online survey, it should be made clear to the participants at which point they can no longer withdraw their data (e.g. once they have pressed submit, when they put the paper based survey in the return box). It should also be clear what will happen to any data collected up to the point of withdrawal (e.g. if it will be retained.

What will happen if I take part?

- How long will the participant be expected to be involved?
    - The input of the participant will be required once, and this will be a process which is expected to take maximum 20 minutes.
- How often will the participants meet the researcher/s if more than once?
    - One response from each participant is needed
- What exactly will happen?
    - Participants would be able to test out the application then provide feedback on the product. This would be in the form of a questionnaire that should last approximately 10 minutes. When the participant is testing out the product, there may be a form which they will need to fill in to register on the application but note that they are not obliged to enter their real personal information as this is just a test and would be provided with different generated identities. It will be anonymous.
- What is the research method used?
    - The method used is in the form of a questionnaire which will include open and closed questions will be asked
- Where is the research taking place? o All research conducted will be taking place online
- What do I have to do if I take part?
    - If the participant decides to take part, they will be asked to test out the product and then follow through with a questionnaire on the application. This can be conducted online and there will be no need to meet in-person.

What will happen if the participant no longer wants to continue?

If the participant wishes to no longer take part in the research, they can withdraw at any time and all information/ responses provided will be disregarded.

Who has reviewed the study?

This study has been approved by City, University of London Ethics Committee.

What if there is a problem?

If you have any problems, concerns or questions about this study, you should ask to speak to a member of the research team. If you remain unhappy and wish to complain formally, you can do this through City's complaints procedure. To complain about the study, you need to phone 020 7040 3040. You can then ask to speak to the Secretary to Senate Research Ethics Committee and inform them that the name of the project is [name of project] You can also write to the Secretary at:

Anna Ramberg
Research Integrity Manager
City, University of London, Northampton Square
London, EC1V 0HB
Email: Anna.Ramberg.1@city.ac.uk

Thank you for taking the time to read this information sheet.

Participant Consent Form

Name of principal investigator/researcher

Suad Warsame

REC reference number

Title of study

Protyes – A mobile application that manages the protests in your area

		Please tick or initial box
1	I confirm that I have read and understood the participant information dated 13th of February 2022 for the above study. I have had the opportunity to consider the information and ask questions which have been answered satisfactorily.	
2.	I understand that my participation is voluntary and that I am free to withdraw without giving a reason without being penalised or disadvantaged.	
3.	I understand that I will be able to withdraw my data up to any time from publication.	
4.	I agree to take part in the above study.	

Name of Participant	Signature	Date
Name of Researcher	Signature	Date

Questionnaire

1) On the scale of 1-10, how easy was it to navigate around the application?
2) Have you been to a protest or demonstration yourself?
3) If yes, were you the organizer of the event?
4) If yes, how easy was this app to organize an event?
5) What is your opinion on protests and demonstrations in general?
6) When using the map, did you manage to easily understand the exact details of the protest that is upcoming, e.g where it is starting and ending, the time and date?
7) How do you find out when a protest is happening in your area?
8) How does an application solely for protests and demonstrations differ to where you obtain your information about protests?
9)

References:

Harding, A., 2022. *South Africa riots: The inside story of Durban's week of anarchy*. [online] BBC News. Available at: <https://www.bbc.co.uk/news/world-africa-57996373> [Accessed 3 February 2022].

Ico.org.uk. 2022. *Guide to the UK General Data Protection Regulation (UK GDPR)*. [online] Available at: <https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-thegeneral-data-protection-regulation-gdpr/> [Accessed 4 February 2022].

Simmons, M., 2022. *The 100-Hour Rule: Forgotten Study Shows How You Can Become WorldClass In 100 Hours*. [online] Medium. Available at: <https://medium.com/acceleratedintelligence/the-100-hour-rule-forgotten-study-shows-how-you-can-become-world-class-in100-hours-ae2f94cc2fb0> [Accessed 4 February 2022].

APPENDIX B: REUSE SUMMARY

The authors legitimate source code comprises 1670 lines of the total code in this project. The table below shows the directories containing reused code and the number of reused codes implemented by the author.

Directory	**Number of Lines**
Intro view	0
Login view	35
adminLogin	8
Firebase Manager	11
DatabaseModel	12
HexColorExtension	27
HomePageView	5
NextEvent	35
ProfileView	0
AdminPage	5
ProtestInfo	4
SubmitInfo	0
AdminContentView	0
UserContentView	0
ImagePicker	50

The author utilised videos such as:

https://www.youtube.com/watch?v=NLOKRKvnHCo&list=LL&index=8&t=522s - **Sign out and sign in feature**

https://www.youtube.com/watch?v=QhLe6ILE_XA&list=LL&index=17&t=630s – **Blog app used to inspire the next event page**

https://www.youtube.com/watch?v=Scver650hQc&list=LL&index=25&t=379s - **Custom map annotation**

https://www.youtube.com/watch?v=TgvYFfCjDMo&list=LL&index=24 - **Navigation bar**

https://www.youtube.com/watch?v=CHFlWHPvY_M&list=LL&index=19 - **Main video used for market and map annotation.**

https://www.youtube.com/watch?v=H6pmm62axCg&list=LL&index=20&t=180s - **More help on marker kit**

https://www.youtube.com/watch?v=89wy2Ilb-FQ&list=LL&index=16&t=517s - **How to fetch and save data to Firebase Firestore**

https://www.youtube.com/watch?v=f6u3AnOKZd0&list=LL&index=14&t=227s - **Helped with implementing live protest tracker.**

https://www.youtube.com/watch?v=3-yQeAf3bLE&list=LL&index=10&t=5s - **Part 2 of helping with implementing live protest tracker.**

The author utilised websites such as:

https://stackoverflow.com/ - **To help with errors along the way**

https://developer.apple.com/ - **To gain background knowledge on Swift**

APPENDIX C: REQUIREMENTS

List of Requirements

1. The user should be able to register by entering an email address and a password
2. The user should be able to log into their account
3. The user should be able to login and submit protest details
4. The user should be able to view the contents of the Next Events page once they logged in
5. The admin should be able to submit protest information on the admin page, so it is visible on the Live Map
6. The admin should be able to update the events on the admin page, so it is visible on the Next Events page
7. The user should be able to log out of their account
8. Details approved by the admin and created on the map view as an annotation should be an instant change once the 'update map' button is clicked.
9. Only the right email and password combination should grant the user access to the application.
10. The user should be able to easily navigate through the application.

Prioritisation of Requirements

Keys:
FR - Functional Requirements
NFR - Non-functional Requirements

Green – High Priority
Amber – Medium Priority
Red – High Priority

Requirement Number	Requirement Type
1	FR
2	FR
3	FR
4	FR
5	FR
6	FR
7	FR
8	NFR
9	NFR
10	NFR

APPENDIX D: VOLERE TEMPLATES

Functional Requirement 1:

Requirement ID: 1	**Requirement Type: Functional**
Description: The user should be able to register by entering an email address and a password	
Rationale: This will allow the users to provide their email address and password so they can register	
Originator: Project manager	
Fit Criteria: Once the user enters their email and password, they should be able to see the message that an account is created	
Customer Satisfaction: 5	**Customer Dissatisfaction:** 5
Priority: High	**Conflicts:** None
Supporting Material: None	Volere Source: Atlantic Systems Guild
History: None	

Functional Requirement 2:

Requirement ID: 2	**Requirement Type: Functional**
Description: The user should be able to log into their account	
Rationale: This will allow the users to log into their account and access the functions of the system	
Originator: Project manager	
Fit Criteria: They can enter the email and password for their account to view the contents related to their account privileges	
Customer Satisfaction: 5	**Customer Dissatisfaction:** 5
Priority: High	**Conflicts:** None
Supporting Material: None	Volere Source: Atlantic Systems Guild
History: None	

Functional Requirement 3:

Requirement ID: 3	**Requirement Type: Functional**
Description: The user should be able to login and submit protest details	
Rationale: This will allow a standard user to have the ability to submit as much details as they can so it can be approved by the admin	
Originator: Project manager	
Fit Criteria: The user will have the ability to log into their account or create one if applicable and navigate to the 'Tell us about a protest tab'. The user can then fill in as much details as possible in order to submit a protest.	
Customer Satisfaction: 4	**Customer Dissatisfaction:** 5
Priority: High	**Conflicts:** None
Supporting Material: None	Volere Source: Atlantic Systems Guild
History: None	

Functional Requirement 4:

Requirement ID: 4	**Requirement Type: Functional**
Description: The user should be able to view the contents of the Next Events page once they logged in	
Rationale: This will allow the users to see the material provided on the Next Events page	
Originator: Project manager	
Fit Criteria: When any user logs into their account, they will have the ability to navigate and click on the Next Events page to view the latest news on protests.	
Customer Satisfaction: 4	**Customer Dissatisfaction:** 2
Priority: Medium	**Conflicts:** None
Supporting Material: None	Volere Source: Atlantic Systems Guild
History: None	

Functional Requirement 5:

Requirement ID: 5	**Requirement Type: Functional**
Description: The admin should be able to submit protest information on the admin page, so it is visible on the Live Map	
Rationale: This will allow the users to see the protests annotated on the map	
Originator: Project manager	
Fit Criteria: When the admin logs into their account, there is a feature which is solely available for the admin account, and they will have the ability to verify and submit the information that was provided under the protestinfo to be displayed on the Live Map	
Customer Satisfaction: 4	**Customer Dissatisfaction:** 5
Priority: High	**Conflicts:** None
Supporting Material: None	Volere Source: Atlantic Systems Guild
History: None	

Functional Requirement 6:

Requirement ID: 6	**Requirement Type: Functional**
Description: The admin should be able to update the events on the admin page, so it is visible on the Next Events page	
Rationale: This will allow the users to see the updated information on the Next Event page	
Originator: Project manager	
Fit Criteria: When the admin logs into their account, there is a feature which is solely available for the admin account, and they will have the ability to enter the latest news which will be displayed on the Next Events page visible for all users to see	
Customer Satisfaction: 4	**Customer Dissatisfaction:** 5
Priority: Medium	**Conflicts:** None
Supporting Material: None	Volere Source: Atlantic Systems Guild
History: None	

Functional Requirement 7:

Requirement ID: 7	**Requirement Type: Functional**
Description: The user should be able to log out of their account	
Rationale: This will allow the users to log out of their account and return to the login page	
Originator: Project manager	
Fit Criteria: When the user logs out, they are able to log back into their account or if they have access to the admin credentials, can log into that	
Customer Satisfaction: 5	**Customer Dissatisfaction:** 3
Priority: Low	**Conflicts:** None
Supporting Material: None	Volere Source: Atlantic Systems Guild
History: None	

Non- Functional Requirement 8:

Requirement ID: 8	**Requirement Type: Non- Functional**
Description: Details approved by the admin and created on the map view as an annotation should be an instant change once the 'update map' button is clicked.	
Rationale: This will allow the users to see the most immediate changes, so they know whether to attend the upcoming protest or not.	
Originator: Project manager	
Fit Criteria: When an admin updates the details for an upcoming protest, the results should be instant, and a point of interest should be called onto the map for all users.	
Customer Satisfaction: 5	**Customer Dissatisfaction:** 4
Priority: Medium	**Conflicts:** None
Supporting Material: None	Volere Source: Atlantic Systems Guild
History: None	

Non- Functional Requirement 9:

Requirement ID: 9	**Requirement Type: Non- Functional**
Description: Only the right email and password combination should grant the user access to the application.	
Rationale: The authenticated users should have access to the system only	
Originator: Project manager	
Fit Criteria: When a user tries to log in by entering their details, the database checks this and if it does not match, then an error message is displayed and the user will have to try again	
Customer Satisfaction: 5	**Customer Dissatisfaction:** 5
Priority: High	**Conflicts:** None
Supporting Material: None	Volere Source: Atlantic Systems Guild
History: None	

Non- Functional Requirement 10:

Requirement ID: 10	**Requirement Type: Non- Functional**
Description: The user should be able to easily navigate through the application.	
Rationale: The user should not have difficulty in clicking from one tab to another and the contents should appear immediately	
Originator: Project manager	
Fit Criteria: The author tests this by allowing the user try out the application and provide feedback through the usability testing questionnaire.	
Customer Satisfaction: 4	**Customer Dissatisfaction:** 5
Priority: Low	**Conflicts:** None
Supporting Material: None	Volere Source: Atlantic Systems Guild
History: None	

APPENDIX E: TESTING

User Acceptance Test (UAT)

Use Test Case	Steps	Expected Results
Log in: The user must be able to login their account by entering their email address and password.	1. The user starts the app 2. Click on 'Get started' button then navigate to 'Login' tab 3. Input email. 4. Input Password. 5. Click 'Login' button.	1. Details entered are fetched from Firebase Authentication to check if there is any saved data 2. User proceeds to the homepage if the details match those stored on Firebase Authentication
Create Account: A user should be able to register by entering an email address and password.	1. The user starts the app 2. Click on 'Get started' button then navigate to 'Create Account' tab 3. Input email. 4. Input Password. 5. Click 'Create Account' button.	1. Details are entered and checked against Firebase Authentication to see if it exists 2. Firebase Authentication stores the email address and password onto the database 3. User can click on the 'login' tab to continue to login to their new account
Setup Profile picture: A user can choose a profile picture for the account created	1. The user chooses a profile picture by picking one from the selection given 2. The user then completes the 'Create Account' section	1. Firebase Storage is used to store the user's picture. 2. The image's URL link is retrieved and saved from Firebase Storage. 3. The user's information as well as the image's URL link are stored to Firebase Firestore.
Submit Protest Details: The user should be able to submit details for an upcoming protest	1. The user is logged In 2. The user goes to the 'Tell us about a protest' page 3. The user fills out the protest details 6. The user clicks 'save' to submit protest details	1. The users' details are sent to Firebase Firestore 2. The details are stored in Firebase Storage 3. The fields should automatically be cleared when the data is entered
Add Map annotation: Admin should be able to add map annotations	1. The Admin is logged in 2. The admin navigates to the 'Admin' tab which has the 'Map' screen 3. The admin fills in the details 4. The admin clicks on the 'Update Map' button	1. The details entered by the admin are sent to Firebase Firestore. 2. The details are then extracted by Firestore and displayed on Map 3. The data inputted by the admin is saved in Firebase Storage.

Add Next Event: Admin should be able to add next event onto the next events page	1. The Admin is logged in 2. The admin navigates to the 'Admin' tab which has the 'Next Event' screen 3. The admin fills in the details	1. The details entered by the admin are sent to Firebase Firestore. 2. The details are then extracted by Firestore and displayed on Next Events page
	4. The admin clicks on the 'Update Next Event' button	3. The data inputted by the admin is saved in Firebase Storage.
Logout: The application should allow the user to log out.	1. The user goes onto the 'Profile' page and clicks on 'Sign out' 2. For admins, click on 'More' screen, then 'Profile' and 'Sign out' button	1. The user is taken to the login page. 2. On Firebase Authentication, users' status is changed to logged out.

Test Cases

Test case 1

Objective: Test Get Started button	**Test Number:** 1
Set Up: Ensure the tester (author) is on the launch screen and that they can click on the 'Get Started' button. The author will then be presented with the login screen.	
Expected Results: Transition from the launch screen to the login screen	
Test: The tester will launch the application and then they will click on the 'Get Started' button to be directed to the login page. The button will be tested 3 times to ensure consistency and that it swiftly changes over to the login screen.	
Test Record: Test successful	
Date: 21 April 2022	**Tester:** Suad Warsame
Result: Passed	

Test case 2

Objective: Test Admin button	**Test Number:** 2
Set Up: Ensure the tester (author) is on the launch screen and that they can click on the 'Admin' button. The author will then be presented with the admin login screen. The button will be tested 3 times to ensure consistency and that it swiftly migrates to the login screen.	
Expected Results: Transition from the launch screen to the login screen	
Test: The tester will launch the application and then they will click on the 'Admin' button to be directed to the admin login page. The button will be tested 3 times to ensure consistency and that it swiftly changes over to the login screen.	
Test Record: Test successful	
Date: 21 April 2022	**Tester:** Suad Warsame
Result: Passed	

Test case 3

Objective: Test Live Map Marker button	**Test Number:** 3
Set Up: Ensure the tester (author) is on the Live Map and that they can click on the red marker on the map. The author will then be presented with the information of the protests. 3 different markers will be tested 3 times to ensure consistency and that it swiftly migrates to the protest information screen.	
Expected Results: Correct point is added to the Live Map No time delay when the marker is pressed	
Test: The tester will launch the application and login. Once logged in, the tester is presented with the Live Map page (homepage) and clicks on a marker to present the tester with the information of the protest. 3 buttons will be tested 3 times to ensure consistency and that it swiftly changes over to the protest information screen.	
Test Record: Test successful	
Date: 21 April 2022	**Tester:** Suad Warsame
Result: Passed	

Test case 4

Objective: Test Update Next Event button	**Test Number:** 4
Set Up: Ensure the tester (author) is on the Admin account and that they can navigate to Admin page. The 'Update Next Button' should be available once on the Admin page. Information will be entered 3 times and the button will be tested each of these times to ensure consistency and that it swiftly populates the contents in the Next Events page.	
Expected Results: Correct information is populated through the Next Events page No time delay when the button is pressed, the information should appear straight away on the Next Events page	
Test: The tester will launch the application and then they will click on the 'Admin' button to be directed to the admin login page. One on the admin page, the tester can enter information on the next event and proceed to click on the 'Update Next Event' button. The button will be tested 3 times to ensure consistency and that it swiftly populates the contents in the Next Events page.	
Test Record: Test unsuccessful	
Date: 21 April 2022	**Tester:** Suad Warsame
Result: Passed	

Test case 5

Objective: Test Update Map button	**Test Number:** 5
Set Up: Ensure the tester (author) is on the Admin account and that they can navigate to Admin page. The 'Update Map' should be available once on the Admin page. Information will be entered 3 times and the button will be tested each of these times to ensure consistency and that it swiftly populates the contents on the Live Map.	
Expected Results: Correct point is added to the Live Map No time delay when the button is pressed, the information should appear straight away on the Live Map	
Test: The tester will launch the application and then they will click on the 'Admin' button to be directed to the admin login page. One on the admin page, the tester can enter annotations on the Live and proceed to click on the 'Update Map' button. The button will be tested 3 times to ensure consistency and that it swiftly populates the contents on the Live Map	
Test Record: Test successful	
Date: 21 April 2022	**Tester:** Suad Warsame
Result: Passed	

Test case 6

Objective: Test Sign out button	**Test Number:** 6
Set Up: Ensure the tester (author) is signed into the application and on the profile page. The tester is then presented with the 'Sign out' button which will be tested 3 times to ensure consistency and that it swiftly returns to the log in screen.	
Expected Results: Transition from profile page to login screen No time delay when the button is pressed	
Test: The tester will launch the application and login. Once logged in, the tester navigates to the profile page and the 'Sign out' button should be present. The button will be tested 3 times to ensure consistency and that it swiftly returns to the log in screen.	
Test Record: Test successful	
Date: 21 April 2022	**Tester:** Suad Warsame
Result: Passed	

Test case 7

Objective: Test Log in button	**Test Number:** 7
Set Up: Ensure the tester (author) has launched the application and has clicked 'Get Started' button. This should present the tester with the login screen with the login fields. The author will then do 3 tests where the 'Log in' button is pressed each time to ensure consistency and that it swiftly migrates to the home page.	
Expected Results: Transition from the log in screen to the homepage No time delay when the button is pressed	
Test: The tester will launch the application and click on the 'Get Started' button. Once the log in view is present, the author will enter the log in details and click on the 'Log in' button. The button will be tested 3 times to ensure consistency and that it swiftly c migrates to the home page.	
Test Record: Test successful	
Date: 21 April 2022	**Tester:** Suad Warsame
Result: Passed	

Test case 8

Objective: Test Create Account button	**Test Number:** 8
Set Up: Ensure the tester (author) has launched the application and has clicked 'Get Started' button. This should present the tester with the create account screen with the registration fields. The author will then do 3 tests where the 'Create Account' button is pressed each time to ensure consistency and that it swiftly migrates to the home page.	
Expected Results: Transition from the log in screen to the homepage No time delay when the button is pressed	
Test: The tester will launch the application and click on the 'Get Started' button. Once the create account view is present, the author will enter the registration details and click on the 'Create Account' button. The button will be tested 3 times to ensure consistency and that it swiftly migrates to the home page.	
Test Record: Test successful	
Date: 21 April 2022	**Tester:** Suad Warsame
Result: Passed	

Test case 9

Objective: Test back button on the Live Map	**Test Number:** 9
Set Up: Ensure the tester (author) is on the Live Map and that they can click on the red marker on the map. The author will then be presented with the information of the protests. There is a back arrow which is located on the top left of the screen and the author conducts 3 tests to ensure consistency and that it swiftly migrates to the Live Map view	
Expected Results: Transition from the protest information to Live Map No time delay when the button is pressed	
Test: Ensure the tester (author) is logged into the account and is present on the Live Map and that they can click on the red marker on the map. The author will then be presented with the information of the protests. There is a back arrow which is located on the top left of the screen and the author conducts 3 tests to ensure consistency and that it swiftly migrates to the Live Map view	
Test Record: Test successful	
Date: 21 April 2022	**Tester:** Suad Warsame
Result: Passed	

Tester Name:

User Testing Questionnaire

Rate the following:

	Strongly Disagree				Strongly Agree
1. I attend protests more than once a year	1	2	3	4	
2. It was easy to navigate around the app	1	2	3	4	
3. Protyes informs you of upcoming protests	1	2	3	4	
4. The design was very basic	1	2	3	4	
5. I would always use the app to track protests	1	2	3	4	
6. There is too much inconsistency	1	2	3	4	
7. Easy to add protest submissions	1	2	3	4	

Please answer these questions below:

1. How do you find out when a protest is happening in your area?

2. What features did you like and why?

3. What features did you not like and why?

4. What features could be improved?

Rate the following:

Design: /5 **Usability**: /5 **Colours**: /5 **Text Size**: /5 **Features**: /5

Tester Name: Sihaam Sheikh

Usability Questionnaire 1:

Rate the following:

	Strongly Disagree				Strongly Agree
8. I attend protests more than once a year	1	2	3	4	
9. It was easy to navigate around the app					X
10. Protyes informs you of upcoming protests					X
11. The design was very basic	X				
12. I would always use the app to track protests				X	
13. There is too much inconsistency	X				
14. Easy to add protest submissions					X

Please answer these questions below:

5. How do you find out when a protest is happening in your area?

Usually go on social media and find information through friend who share posts, pages that I follow might reshare posts as well

6. What features did you like and why?

I liked the interactive map and how you can check details on protests, it allows you to be organised and up to date

7. What features did you not like and why?

No dislikes, the application was very good for a computer science student with basic coding knowledge

8. What features could be improved?

Maybe have a notification feature that shows similar protests in the area so you can be in tune with them as well

Rate the following:

Design: 5/5 **Usability**: 5/5 **Colours**: 4/5 **Text Size**: 5/5 **Features**: 5/5

Tester Name: Samiira Mohamed

Usability Questionnaire 2:

Rate the following:

	Strongly Disagree				Strongly Agree
15. I attend protests more than once a year §	1	2	3	4	
16. It was easy to navigate around the app					X
17. Protyes informs you of upcoming protests					X
18. The design was very basic	X				
19. I would always use the app to track protests					X
20. There is too much inconsistency	X				
21. Easy to add protest submissions					X

Please answer these questions below:

9. How do you find out when a protest is happening in your area?

Through friends, social media

10. What features did you like and why?

The concept of the user submitting protests was cool, and the fact the admins display the info on the map is even better!

11. What features did you not like and why?

Nothing, the app is an incredible concept

12. What features could be improved?

Have a subscription page instead of a next page where you can subscribe to movements, seeing what latest protests are

Rate the following:

Design: 5/5 **Usability**: 5/5 **Colours**: 5/5 **Text Size**: 5/5 **Features**: 5/5

Tester Name: Mark Barber

Usability Questionnaire 3:

Rate the following:

	Strongly Disagree				Strongly Agree
22. I attend protests more than once a year	1	2	3	4	
23. It was easy to navigate around the app					X
24. Protyes informs you of upcoming protests				X	
25. The design was very basic		X			
26. I would always use the app to track protests			X		
27. There is too much inconsistency	X				
28. Easy to add protest submissions					X

Please answer these questions below:

13. How do you find out when a protest is happening in your area?

Traditional way through socials

14. What features did you like and why?

The interactive map and the ability to click on a pointer to view more information

15. What features did you not like and why?

That the coordinates must manually be imputed, should be an automatic thing

16. What features could be improved?

Maybe have the application as an integration to an app most people have, like Waze, I have too many apps on my phone

Rate the following:

Design: 5/5 **Usability**: 4/5 **Colours**: 4/5 **Text Size**: 5/5 **Features**: 4/5

Tester Name: Emmanuel Ashley

Usability Questionnaire 4:

Rate the following:

Strongly
Strongly

29. I attend protests more than once a year

	1	2	3	4
30. It was easy to navigate around the app				X
31. Protyes informs you of upcoming protests				X
32. The design was very basic	X			
33. I would always use the app to track protests				X
34. There is too much inconsistency	X			
35. Easy to add protest submissions				X

Please answer these questions below:

17. How do you find out when a protest is happening in your area?

I don't really attend protests and if I do, I find out through friends

18. What features did you like and why?

The Next events page, its informative and a unique feature to have

19. What features did you not like and why?

Not really a bad thing but there were only two feeds in the Next Events page, more is needed

20. What features could be improved?

The next events page should have more interesting posts!

Rate the following:

Design: 5/5 **Usability**: 5/5 **Colours**: 5/5 **Text Size**: 5/5 **Features**: 4/5

Tester Name: Hager Beyan

Usability Questionnaire 5:

Rate the following:

Strongly
Strongly

36. I attend protests more than once a year

1	2	3	4	
				X

37. It was easy to navigate around the app

1	2	3	4	
				X

38. Protyes informs you of upcoming protests

1	2	3	4	
	X			

39. The design was very basic

1	2	3	4	
			X	

40. I would always use the app to track protests

1	2	3	4	
X				

41. There is too much inconsistency

1	2	3	4	
				X

42. Easy to add protest submissions

Please answer these questions below:

21. How do you find out when a protest is happening in your area?

On places like Instagram and Snapchat, sometimes Tiktok

22. What features did you like and why?

The admin page, how they can easily add protests to reflect on the map

23. What features did you not like and why?

The manual entry of the location, it was very long choosing a place

24. What features could be improved?

Some sort of subscription to channels or like a tweet and communications page

Rate the following:

Design: 4/5 **Usability**: 4/5 **Colours**: 4/5 **Text Size**: 5/5 **Features**: 4/5

Tester Name: Gulled Siman

Usability Questionnaire 6:

Rate the following:

	Strongly Disagree				Strongly Agree
43. I attend protests more than once a year	1	2	3	4	
44. It was easy to navigate around the app					X
45. Protyes informs you of upcoming protests					X
46. The design was very basic	X				
47. I would always use the app to track protests			X		
48. There is too much inconsistency	X				
49. Easy to add protest submissions					X

Please answer these questions below:

25. How do you find out when a protest is happening in your area?

Social media, extinction rebellion

26. What features did you like and why?

the map that moves around and you can click on points to view the information

27. What features did you not like and why?

No automatic coordinates when submitting a protest, could be tedious, something to check out

28. What features could be improved?

The coordinates, making them populate when you put a location in

Rate the following:

Design: 5/5 **Usability**: 4/5 **Colours**: 5/5 **Text Size**: 5/5 **Features**: 4/5

Tester Name: Sehyr Khan

Usability Questionnaire 7:

Rate the following:

Strongly
Strongly

50. I attend protests more than once a year

	1	2	3	4
51. It was easy to navigate around the app				X
52. Protyes informs you of upcoming protests			X	
53. The design was very basic	X			
54. I would always use the app to track protests			X	
55. There is too much inconsistency	X			
56. Easy to add protest submissions				X

Please answer these questions below:

29. How do you find out when a protest is happening in your area?

Instagram, google, news, facebook

30. What features did you like and why?

the map that shows us the details on the protest once pressed

31. What features did you not like and why?

There is only one admin account that puts info in for protests, what happens if important information is missed, who can verify

32. What features could be improved?

Have a part where you can upload pictures for the protest submission letters, so admin can view letters from officials

Rate the following:

Design: 5/5 **Usability**: 5/5 **Colours**: 5/5 **Text Size**: 5/5 **Features**: 4/5

Tester Name: Sana Khan

Usability Questionnaire 8:

Rate the following:

	Strongly Disagree				Strongly Agree
57. I attend protests more than once a year	1	2	3	4	
58. It was easy to navigate around the app					X
59. Protyes informs you of upcoming protests					X
60. The design was very basic	X				
61. I would always use the app to track protests					X
62. There is too much inconsistency	X				
63. Easy to add protest submissions					X

Please answer these questions below:

33. How do you find out when a protest is happening in your area?

Social media

34. What features did you like and why?

The map was very good, you can even zoom out and check global protests

35. What features did you not like and why?

The next events page was limited to two events, there should be a variety

36. What features could be improved?

The next events page, have a variety of events listed there so user can interact with the multiple content

Rate the following:

Design: 5/5 **Usability**: 5/5 **Colours**: 5/5 **Text Size**: 5/5 **Features**: 5/5

APPENDIX F: CONSENT FORMS

Participant consent form Template

Participant Consent Form

Name of principal investigator/researcher

Suad Warsame

REC reference number

Title of study

Protyes – A mobile application that manages the protests in your area

		Please tick or initial box
1	I confirm that I have read and understood the participant information dated 13th of February 2022 for the above study. I have had the opportunity to consider the information and ask questions which have been answered satisfactorily.	
2.	I understand that my participation is voluntary and that I am free to withdraw without giving a reason without being penalised or disadvantaged.	
3.	I understand that I will be able to withdraw my data up to any time from publication.	
4.	I agree to take part in the above study.	

_____	_____	_____
Name of Participant	Signature	Date
_____	_____	_____
Name of Researcher	Signature	Date

Participant consent form 1:

Participant Consent Form

Name of principal investigator/researcher

Suad Warsame

REC reference number

Title of study

Protyes – A mobile application that manages the protests in your area

		Please tick or initial box
1	I confirm that I have read and understood the participant information dated 13th of February 2022 for the above study. I have had the opportunity to consider the information and ask questions which have been answered satisfactorily.	✓
2.	I understand that my participation is voluntary and that I am free to withdraw without giving a reason without being penalised or disadvantaged.	✓
3.	I understand that I will be able to withdraw my data up to any time from publication.	✓
4.	I agree to take part in the above study.	✓

Sihaam sheikh		01/ 04/22
Name of Participant	Signature	Date
Suad Warsame		01/04/22
Name of Researcher	Signature	Date

Participant consent form 2:

Participant Consent Form

Name of principal investigator/researcher

Suad Warsame

REC reference number

Title of study

Protyes – A mobile application that manages the protests in your area

		Please tick or initial box
1	I confirm that I have read and understood the participant information dated 13th of February 2022 for the above study. I have had the opportunity to consider the information and ask questions which have been answered satisfactorily.	✓
2.	I understand that my participation is voluntary and that I am free to withdraw without giving a reason without being penalised or disadvantaged.	✓
3.	I understand that I will be able to withdraw my data up to any time from publication.	✓
4.	I agree to take part in the above study.	✓

Samira Mohamed		01/ 04/22
Name of Participant	Signature	Date
Suad Warsame		01/04/22
Name of Researcher	Signature	Date

Participant consent form 3:

Participant Consent Form

Name of principal investigator/researcher

Suad Warsame

REC reference number

Title of study

Protyes – A mobile application that manages the protests in your area

		Please tick or initial box
1	I confirm that I have read and understood the participant information dated 13th of February 2022 for the above study. I have had the opportunity to consider the information and ask questions which have been answered satisfactorily.	✓
2.	I understand that my participation is voluntary and that I am free to withdraw without giving a reason without being penalised or disadvantaged.	✓
3.	I understand that I will be able to withdraw my data up to any time from publication.	✓
4.	I agree to take part in the above study.	✓

Mark Barber	M. Barber	01/ 04/22
Name of Participant	Signature	Date
Suad Warsame		01/04/22
Name of Researcher	Signature	Date

Participant consent form 4:

Participant Consent Form

Name of principal investigator/researcher

Suad Warsame

REC reference number

Title of study

Protyes – A mobile application that manages the protests in your area

		Please tick or initial box
1	I confirm that I have read and understood the participant information dated 13th of February 2022 for the above study. I have had the opportunity to consider the information and ask questions which have been answered satisfactorily.	✓
2.	I understand that my participation is voluntary and that I am free to withdraw without giving a reason without being penalised or disadvantaged.	✓
3.	I understand that I will be able to withdraw my data up to any time from publication.	✓
4.	I agree to take part in the above study.	✓

Emmanuel Ashley		01/ 04/22
Name of Participant	Signature	Date
Suad Warsame		01/04/22
Name of Researcher	Signature	Date

Participant consent form 5:

Participant Consent Form

Name of principal investigator/researcher

Suad Warsame

REC reference number

Title of study

Protyes – A mobile application that manages the protests in your area

		Please tick or initial box
1	I confirm that I have read and understood the participant information dated 13th of February 2022 for the above study. I have had the opportunity to consider the information and ask questions which have been answered satisfactorily.	✓
2.	I understand that my participation is voluntary and that I am free to withdraw without giving a reason without being penalised or disadvantaged.	✓
3.	I understand that I will be able to withdraw my data up to any time from publication.	✓
4.	I agree to take part in the above study.	✓

Hager Beyan	[signature]	01/ 04/22
Name of Participant	Signature	Date
Suad Warsame	[signature]	01/04/22
Name of Researcher	Signature	Date

Participant consent form 6:

Participant Consent Form

Name of principal investigator/researcher

Suad Warsame

REC reference number

Title of study

Protyes – A mobile application that manages the protests in your area

		Please tick or initial box
1	I confirm that I have read and understood the participant information dated 13th of February 2022 for the above study. I have had the opportunity to consider the information and ask questions which have been answered satisfactorily.	✓
2.	I understand that my participation is voluntary and that I am free to withdraw without giving a reason without being penalised or disadvantaged.	✓
3.	I understand that I will be able to withdraw my data up to any time from publication.	✓
4.	I agree to take part in the above study.	✓

Gulled Siman	G.SIMAN	01/ 04/22
Name of Participant	Signature	Date
Suad Warsame		01/04/22
Name of Researcher	Signature	Date

Participant consent form 7:

Participant Consent Form

Name of principal investigator/researcher

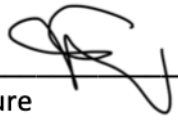
Suad Warsame

REC reference number

Title of study

Protyes – A mobile application that manages the protests in your area

		Please tick or initial box
1	I confirm that I have read and understood the participant information dated 13th of February 2022 for the above study. I have had the opportunity to consider the information and ask questions which have been answered satisfactorily.	✓
2.	I understand that my participation is voluntary and that I am free to withdraw without giving a reason without being penalised or disadvantaged.	✓
3.	I understand that I will be able to withdraw my data up to any time from publication.	✓
4.	I agree to take part in the above study.	✓

Sehyr Khan		01/ 04/22
Name of Participant	Signature	Date
Suad Warsame		01/04/22
Name of Researcher	Signature	Date

Participant consent form 8:

Participant Consent Form

Name of principal investigator/researcher

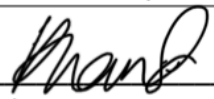
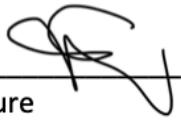
Suad Warsame

REC reference number

Title of study

Protyes – A mobile application that manages the protests in your area

		Please tick or initial box
1	I confirm that I have read and understood the participant information dated 13th of February 2022 for the above study. I have had the opportunity to consider the information and ask questions which have been answered satisfactorily.	✓
2.	I understand that my participation is voluntary and that I am free to withdraw without giving a reason without being penalised or disadvantaged.	✓
3.	I understand that I will be able to withdraw my data up to any time from publication.	✓
4.	I agree to take part in the above study.	✓

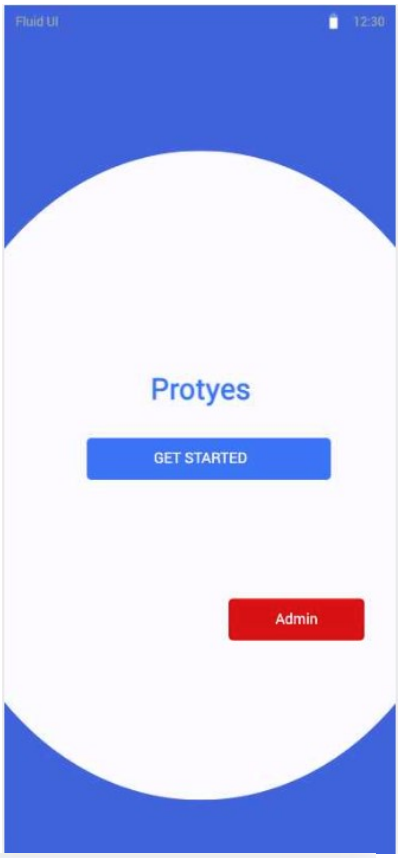
Sana Khan		01/ 04/22
Name of Participant	Signature	Date
Suad Warsame		01/04/22
Name of Researcher	Signature	Date

APPENDIX G: DESIGN

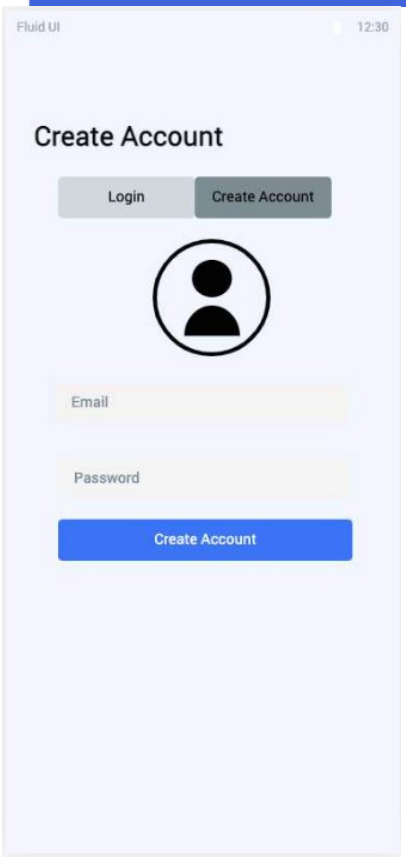
Wireframes (Low fidelity)

Wireframes (High fidelity)

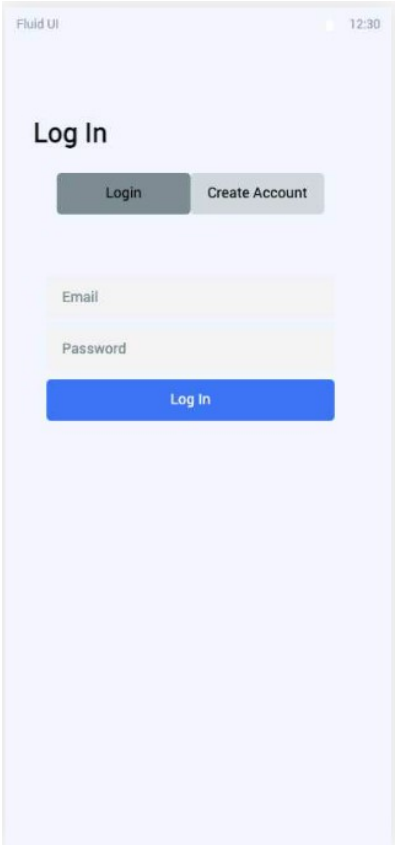
1.1 Initial page



1.2 Create account page

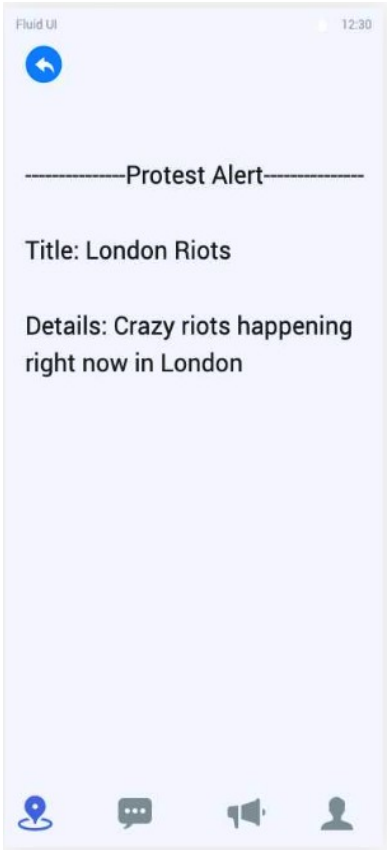


1.2.1 Login page



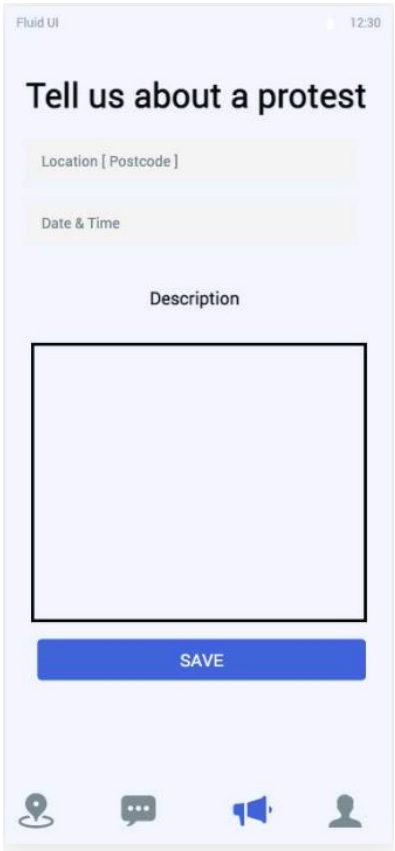
1.3 Map page

1.3.1 Protest Point (Map Page)



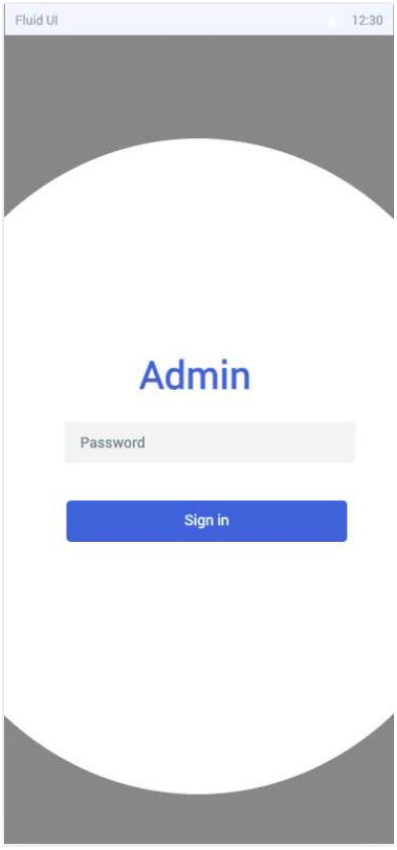
1.4 Next Events page

1.5 Submit Protest page

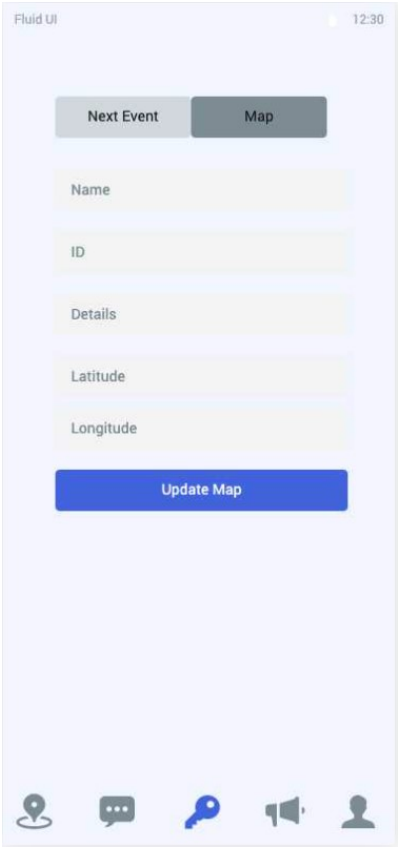


1.6 Profile page (standard user)

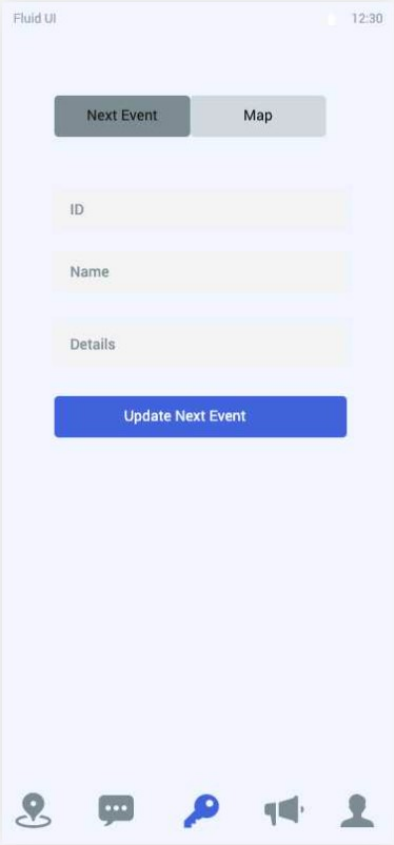
1.7 Admin Login page



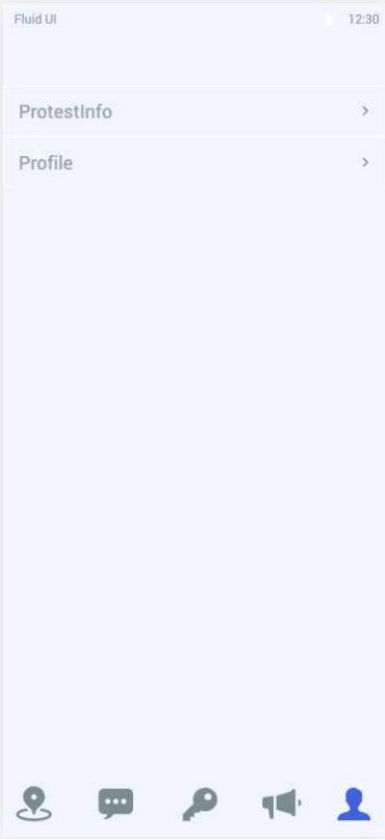
1.8 Admin page (Map)



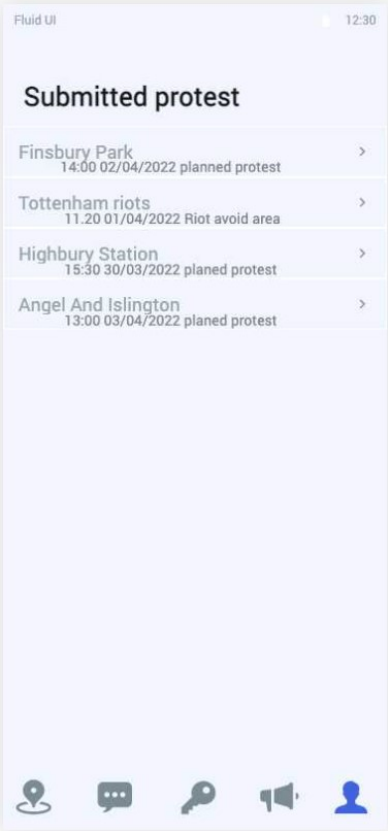
8.1 Admin page (next Event)



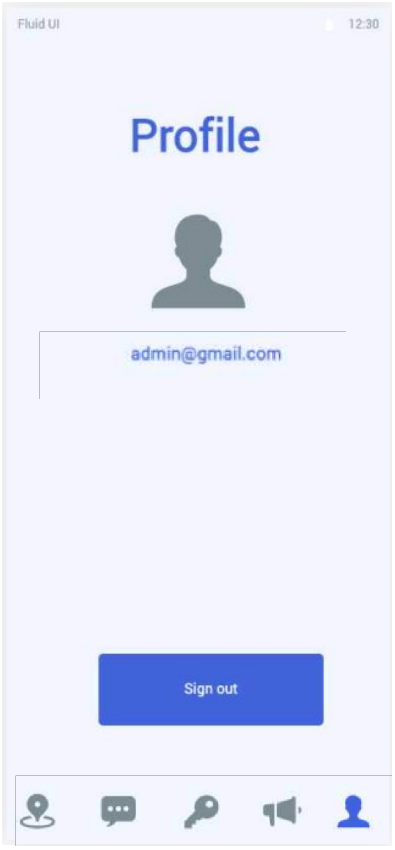
1.9 More page



1.9.1 ProtestInfo page

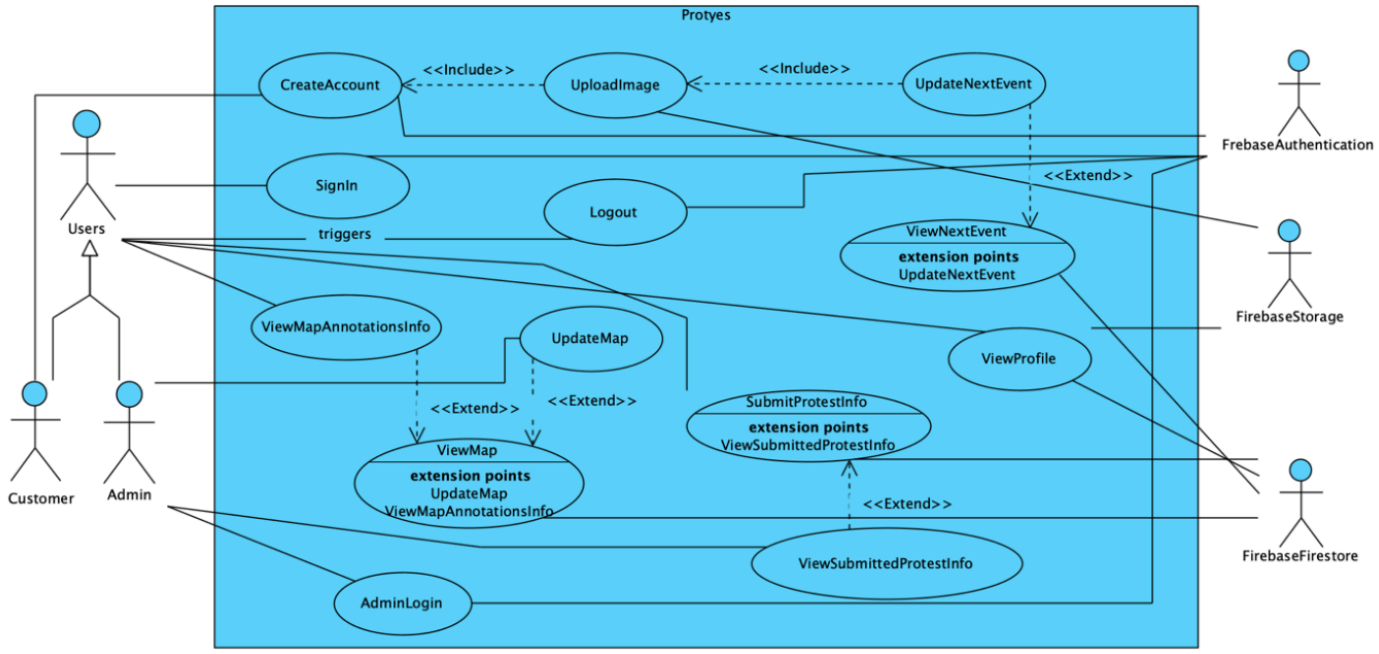


1.9.2 Profile page (admin)



Navigation Architecture Diagram

Use Case Diagram



Use Case Specification

Use case: Login

ID: 1
Brief Description: If a user decides to access the system, for authenticity, they must input their email and password, which will in turn grant them access to their account
Primary Actors: User
Secondary Actors: Firebase Authentication
Preconditions: - Application is operational - The login in screen has opened, and it must be completed
Main Flow: 1. The use case begins when user launches the programme 2. The system prompts the user to input their information, including their email and password 3. The user types in their email 4. The user types in their password 5. The "Log In" button is selected by the user 6. The information is subsequently given to Firebase Authentication for verification 7. If the user's information is fetched by Firebase Authentication, the user moves on to step 9 8. If an error message is returned to the application, the message is shown as an alternate flow (wrong Email, wrong Password) 9. If the user is a regular user 9.1 The user's information is retrieved from Firestore by the system 9.2 The shared preference settings are established by the system 9.3 The user is sent to the Home Page by the system 10. If the user is an administrator 10.1 The user is sent to the administration portal by the system
Postconditions: The user is sent to a page that is suitable for their role.
Alternative Flow: wrong Email wrong Password

Alternative flow: wrongPassword

Alternative Flow: wrongPassword
ID: 1.1
Brief Description: User enters the wrong Password
Primary Actors: User
Secondary Actors: Firebase Authentication
Preconditions: The user's information was submitted to Firebase for authentication
Alternative Flow: 1. This alternate flow begins only if the ‘Log In’ step fails (step 8) 2. "Wrong Password inputted” is the message that is sent to the application
Postconditions: none

Alternative flow: wrongEmail

Alternative Flow: wrongEmail
ID: 1.2
Brief Description: User enters the wrong email
Primary Actors: User
Secondary Actors: Firebase Authentication
Preconditions: The user's information was submitted to Firebase for authentication
Alternative Flow: 1. This alternate flow begins only if the ‘Log In’ step fails (step 8) 2. "Wrong Email inputted” is the message that is sent to the application
Postconditions: none

Use case: createAccount

ID: 2
Brief Description: For identification reasons, when a user wants to access the application, they must create an account using their email
Primary Actors: User
Secondary Actors: Firebase Authentication
Preconditions: - The user is present on the create account page - The application is working
Main Flow: 1. The use case begins when user launches the programme 2. The system prompts the user to input their information, including their email and password 3. The user types in their email 4. The user types in their password 5. The "Create Account" button is selected by the user 6. The information is subsequently given to Firebase Authentication for verification 7. If the user's information is fetched by Firebase Authentication, the user moves on to step 9 8. If an error message is returned to the application, the message is shown as an alternate flow (wrong Email Format, wrong Password Format) 9. If Firebase Authentication accepts the user's information 10. The user utilises the newly created email address and password which is recorded using Firebase authentication 11. The user goes onto the Log in page to proceed with the login and is subsequently sent to the Home Page by the system
Postconditions: The user is sent to the Login screen then homepage
Alternative Flow: wrongEmailFormat wrongPasswordFormat

Alternative flow: wrongEmailFormat

Alternative Flow: wrongEmailFormat
ID: 2.2
Brief Description: User enters the wrong email format
Primary Actors: User
Secondary Actors: Firebase Authentication
Preconditions: The user's information was submitted to Firebase for authentication
Alternative Flow: 1. This alternate flow begins only if the ‘Create Account’ step fails (step 8) 2. "Email address is badly formatted” is the message that is sent to the application
Postconditions: none

Alternative flow: wrongPasswordFormat

Alternative Flow: wrongPasswordFormat
ID: 2.3
Brief Description: User enters the wrong password
Primary Actors: User
Secondary Actors: Firebase Authentication
Preconditions: The user's information was submitted to Firebase for authentication
Alternative Flow: 1. This alternate flow begins only if the ‘Create Account’ step fails (step 8) 2. "Password is badly formatted” is the message that is sent to the application
Postconditions: none

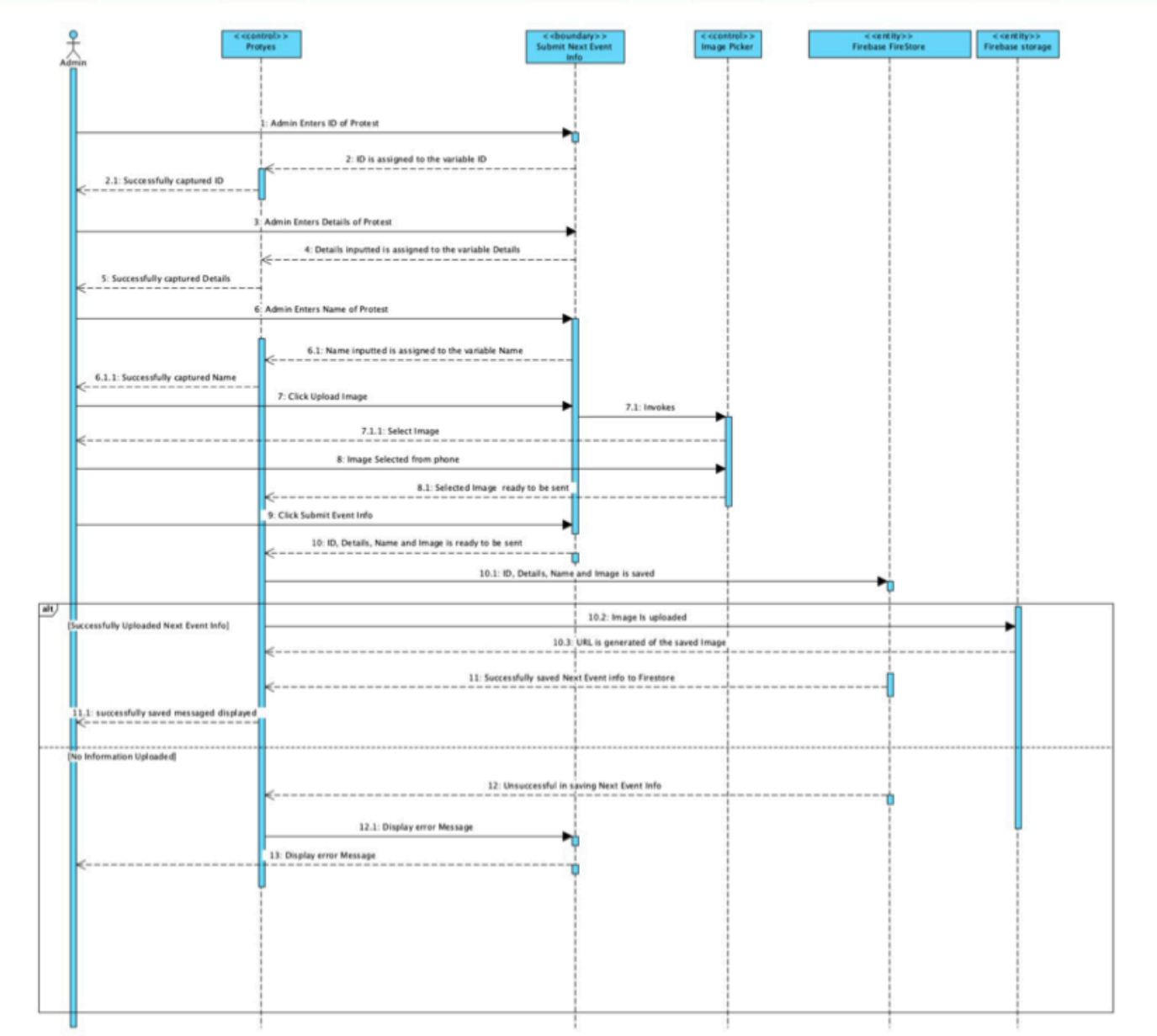
Use Case: Logout

ID: 3
Brief Description: User can log out of their profile by clicking the button available on the profile page
Primary Actors: Users
Secondary Actors: Firebase Authentication
Preconditions: - The user is logged in - The user is on the profile page
Main Flow: 1.The use case starts by the users logging into their account 2. To successfully log out 2.2 If a user is on a standard account, they would navigate to the profile button 2.3 If it is an admin user, they will click on the 'More' page and then profile button 3. The user then proceeds to log out by clicking on the 'logout' button
Postconditions: The user is presented with the launch screen again
Alternative Flow: None

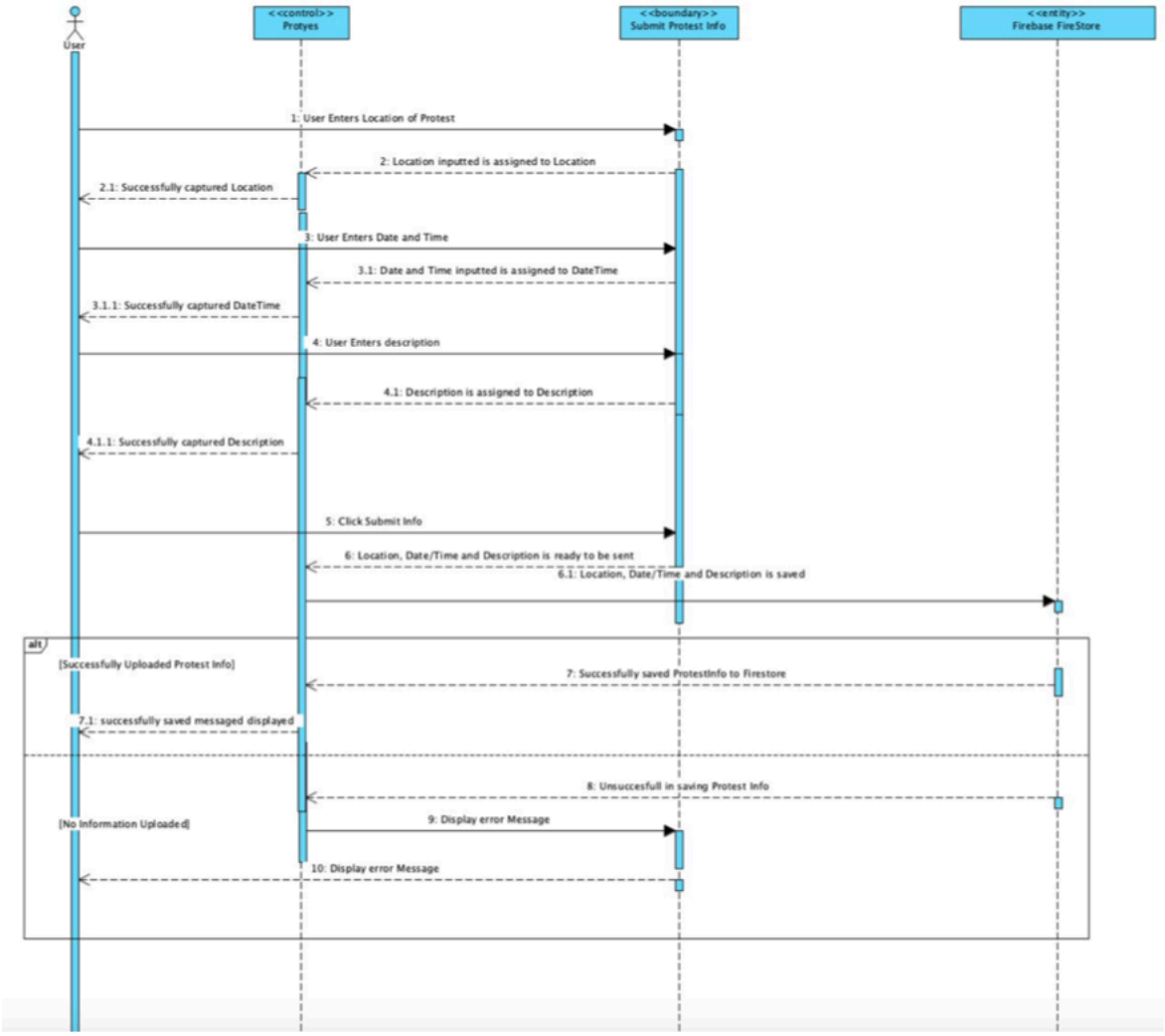
Use case: submitProtestInfo

ID: 4
Brief Description: When a user wants to submit protest details, they will be required to fill out a form
Primary Actors: User
Secondary Actors: Firebase Firestore
Preconditions: The user is logged in The user is on the 'Tell us about a protest' page
Main Flow: 1. The use case starts by the user entering the details of a protest that is upcoming through the designated screen 2. The user enters details such as the location, time, date and a description 3. The data entered is then sent over and uploaded to Firebase Firestore 4. If the information is uploaded successfully to Firestore 4.4 All the fields are cleared 4.5 a message is displayed reading 'Upload successful' 5. If the information is entered incorrectly 5.5 The user will see an error message
Postconditions: The protest information is uploaded to Firestore with the message 'Upload successful'
Alternative Flow: None

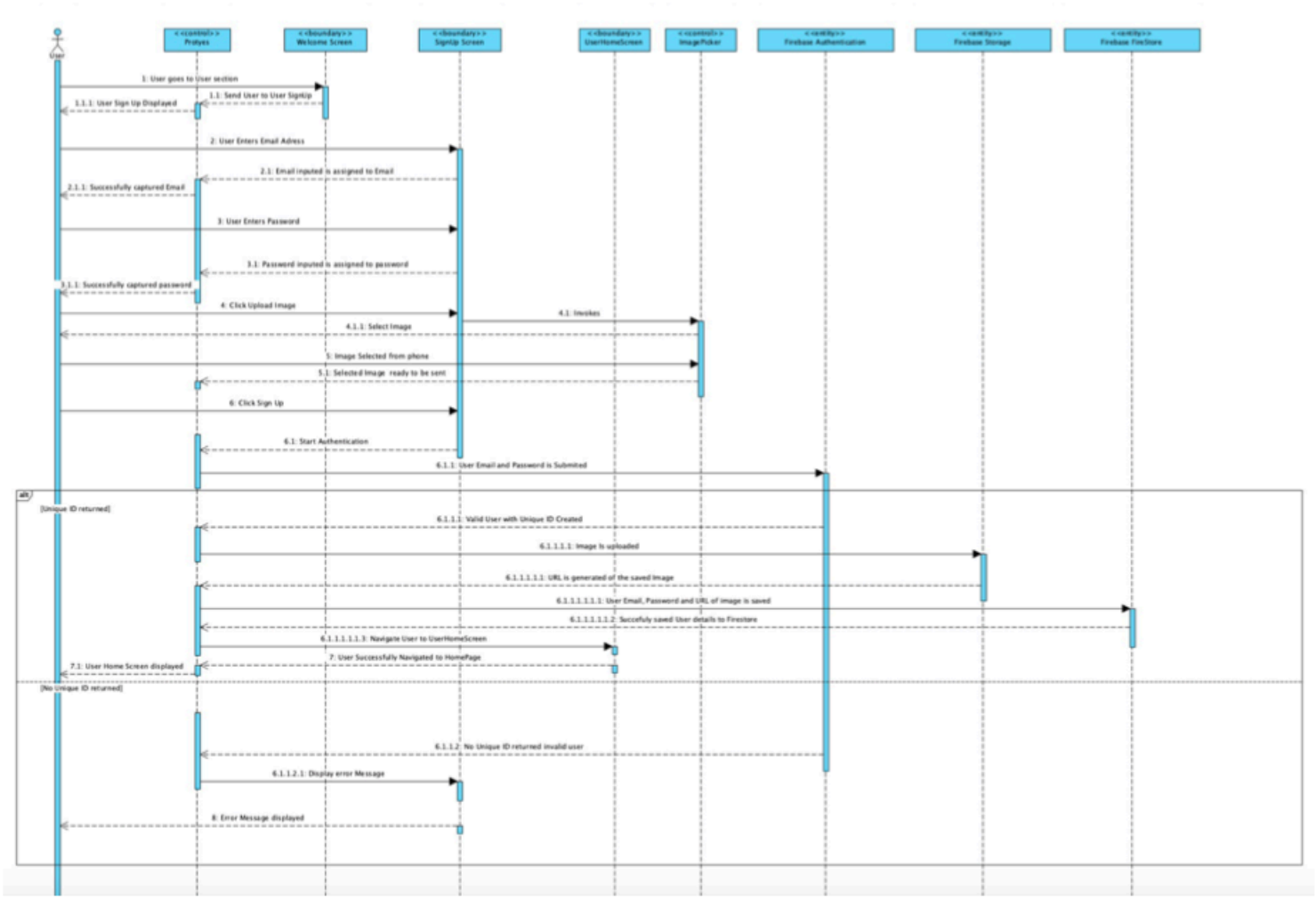
Sequence Diagram
Submit Event Info



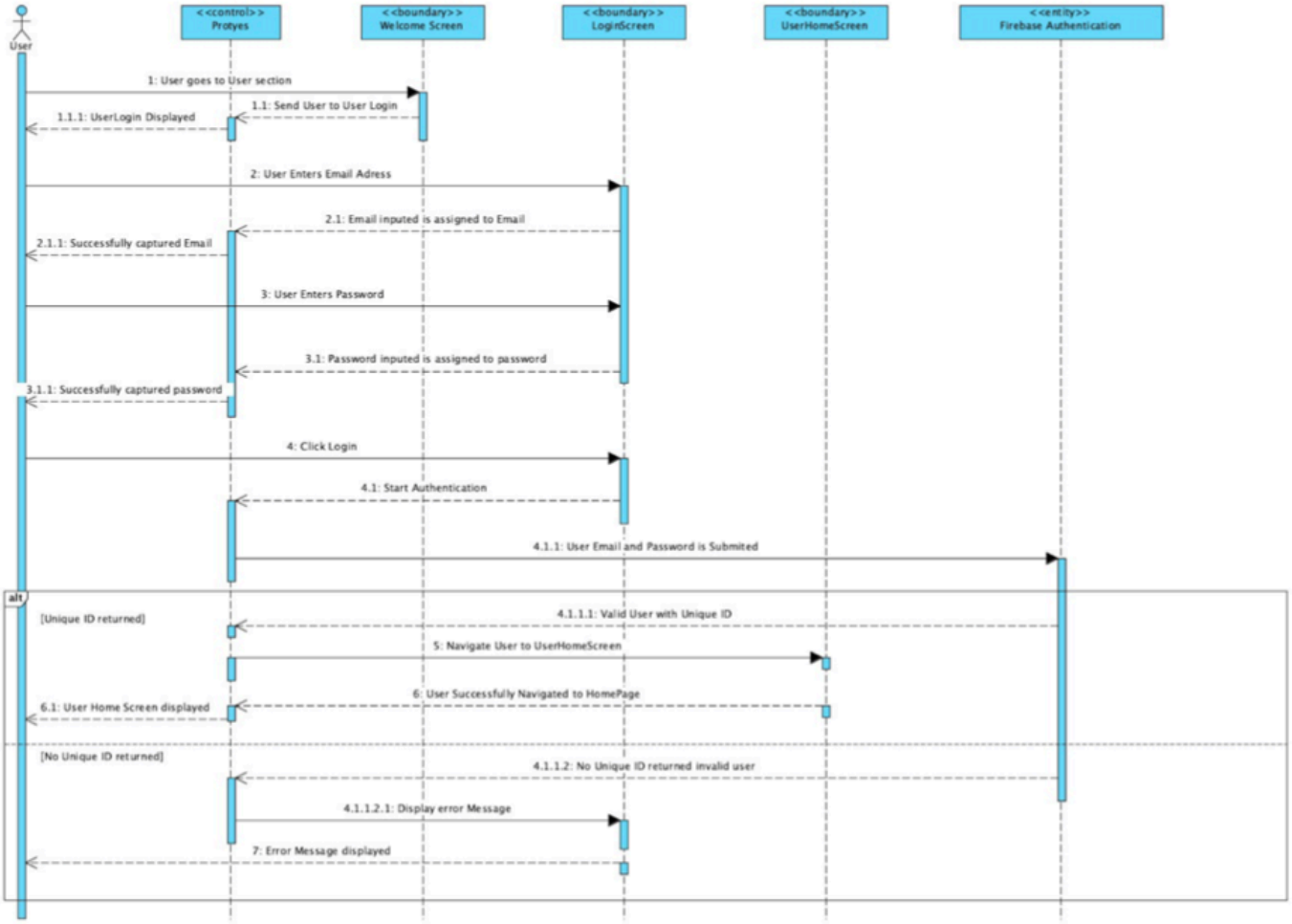
Submit Protest Info



Sign Up

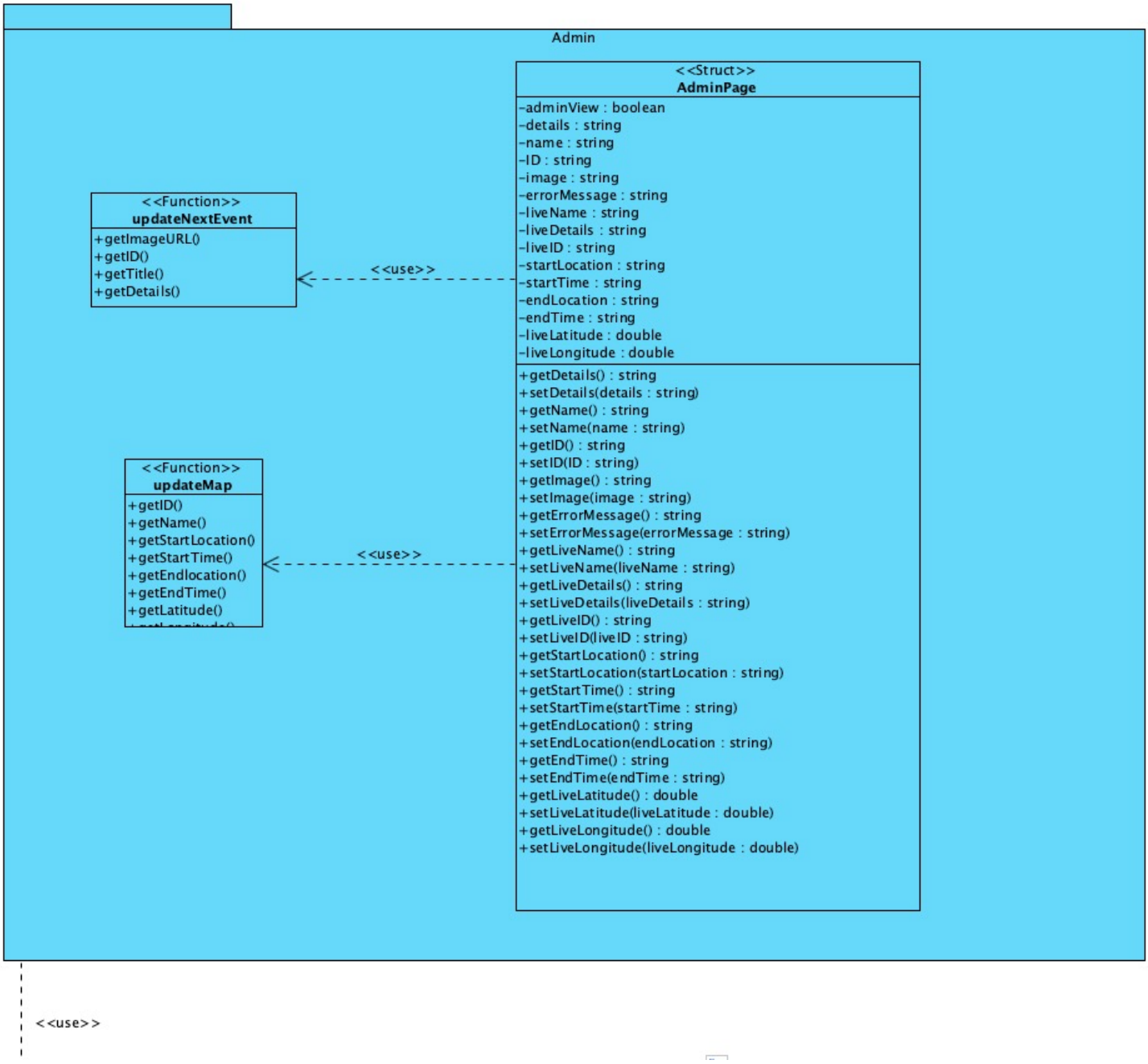

User
<<control>>
Protyes
<<boundary>>
Welcome Screen
<<boundary>>
SignUp Screen
<<boundary>>
UserHomeScreen
<<control>>
ImagePicker
<<entity>>
Firebase Authentication
<<entity>>
Firebase Storage
<<entity>>
Firebase FireStore
1: User goes to User section
1.1: Send User to User SignUp
1.1.1: User Sign Up Displayed
2: User Enters Email Adress
2.1: Email inputed is assigned to Email
2.1.1: Successfully captured Email
3: User Enters Password
3.1: Password inputed is assigned to password
3.1.1: Successfully captured password
4: Click Upload Image
4.1: Invokes
4.1.1: Select Image
5: Image Selected from phone
5.1: Selected Image ready to be sent
6: Click Sign Up
6.1: Start Authentication
6.1.1: User Email and Password is Submited
alt
[Unique ID returned]
6.1.1.1: Valid User with Unique ID Created
6.1.1.1.1: Image is uploaded
6.1.1.1.1.1: URL is generated of the saved Image
6.1.1.1.1.1.1: User Email, Password and URL of image is saved
6.1.1.1.1.1.2: Succesfuly saved User details to Firestore
6.1.1.1.1.1.3: Navigate User to UserHomeScreen
7: User Successfully Navigated to HomePage
7.1: User Home Screen displayed
[No Unique ID returned]
6.1.1.2: No Unique ID returned invalid user
6.1.1.2.1: Display error Message
8: Error Message displayed

Log in

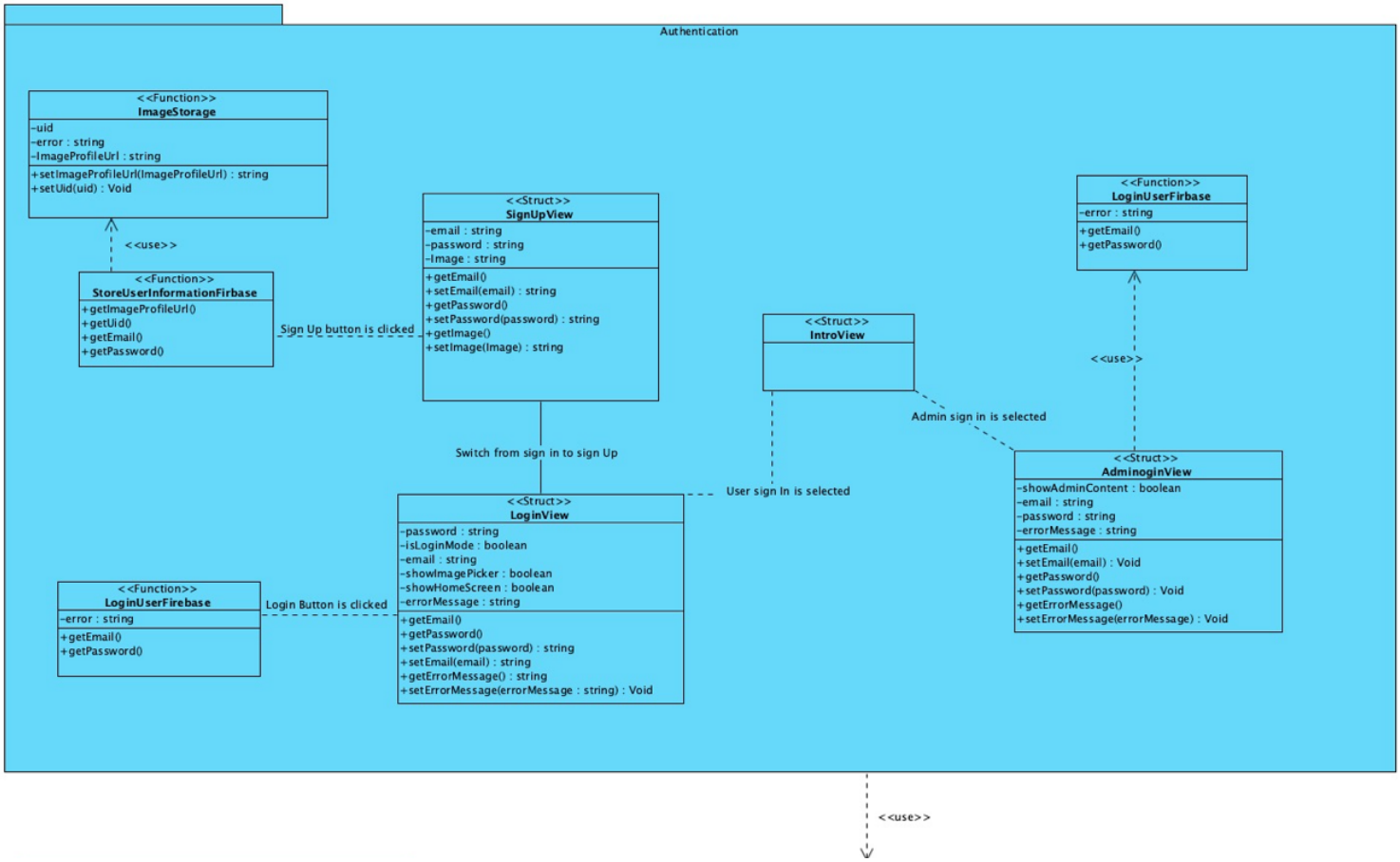

User
<<control>>
Protyes
<<boundary>>
Welcome Screen
<<boundary>>
LoginScreen
<<boundary>>
UserHomeScreen
<<entity>>
Firebase Authentication
1: User goes to User section
1.1: Send User to User Login
1.1.1: UserLogin Displayed
2: User Enters Email Adress
2.1: Email inputed is assigned to Email
2.1.1: Successfully captured Email
3: User Enters Password
3.1: Password inputed is assigned to password
3.1.1: Successfully captured password
4: Click Login
4.1: Start Authentication
4.1.1: User Email and Password is Submited
alt
[Unique ID returned]
4.1.1.1: Valid User with Unique ID
5: Navigate User to UserHomeScreen
6: User Successfully Navigated to HomePage
6.1: User Home Screen displayed
[No Unique ID returned]
4.1.1.2: No Unique ID returned invalid user
4.1.1.2.1: Display error Message
7: Error Message displayed

Class Diagram

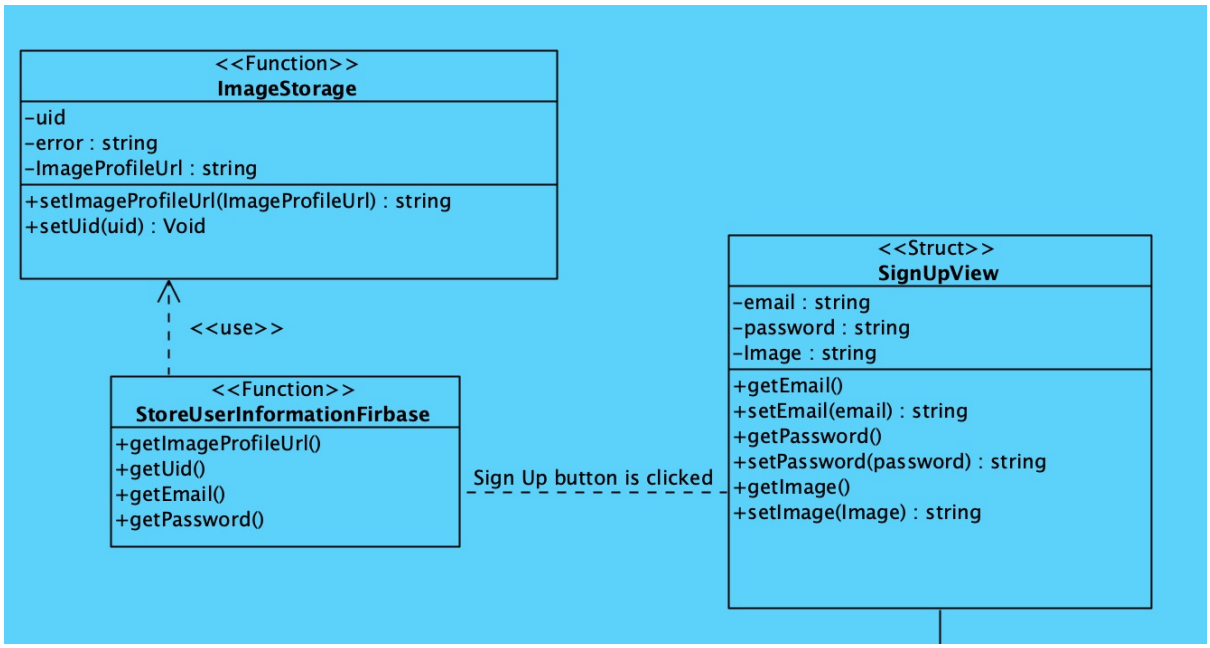
Admin Package



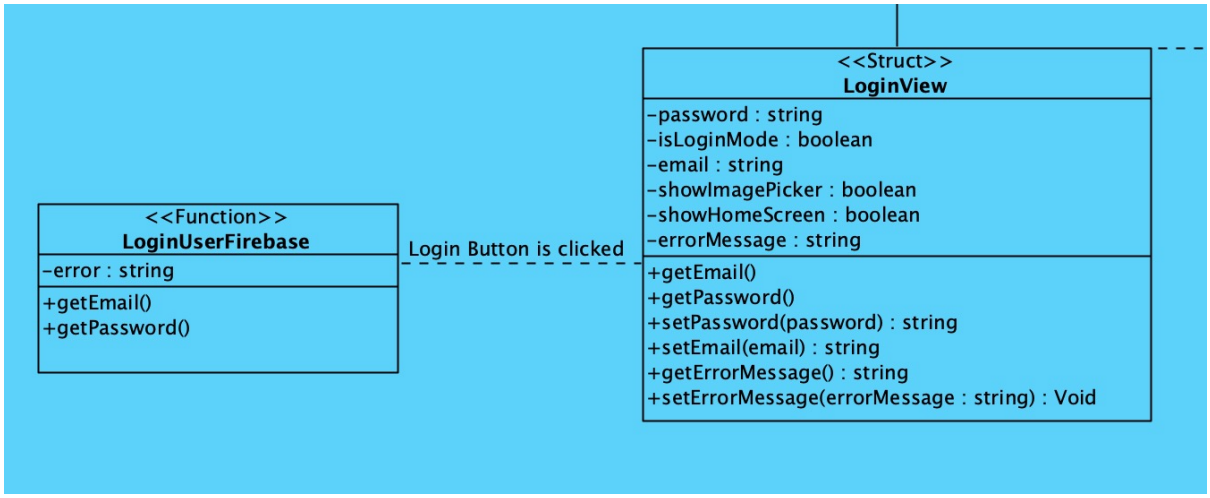
Authentication Package



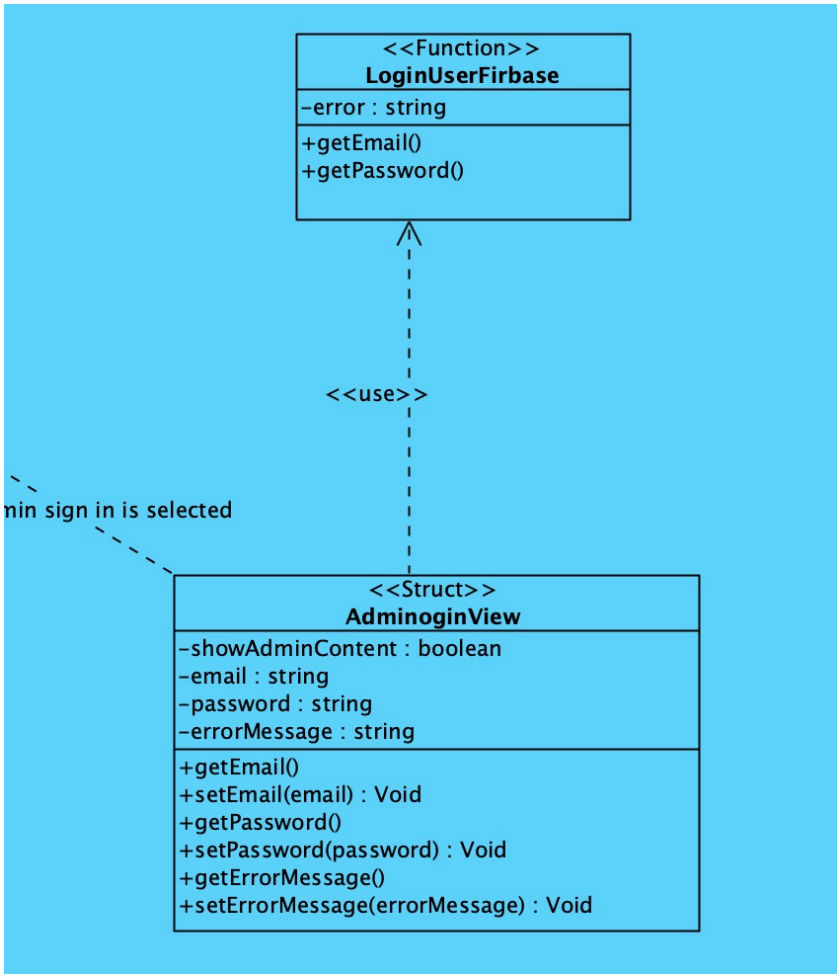
Sign Up Class



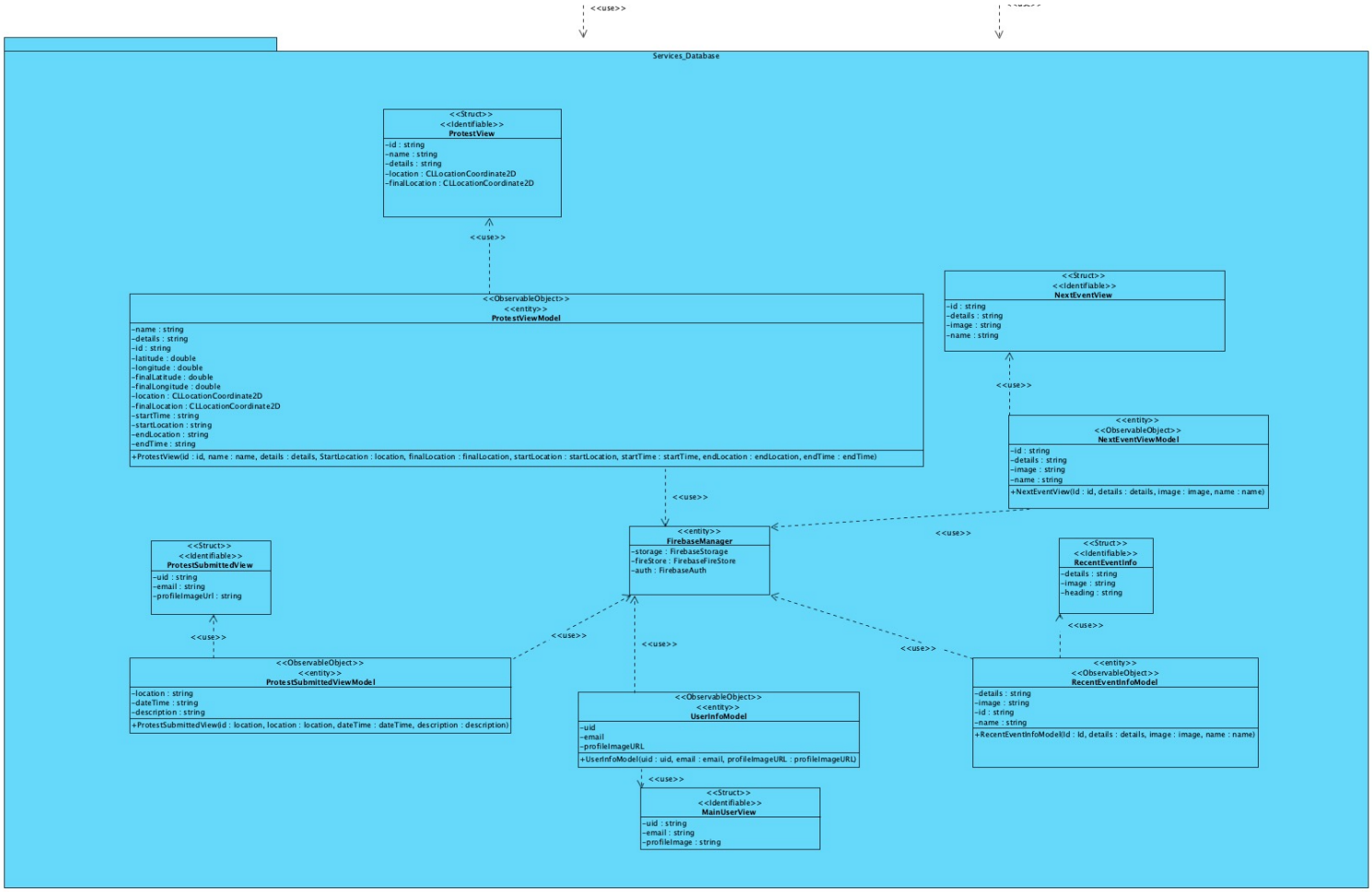
Login Class User



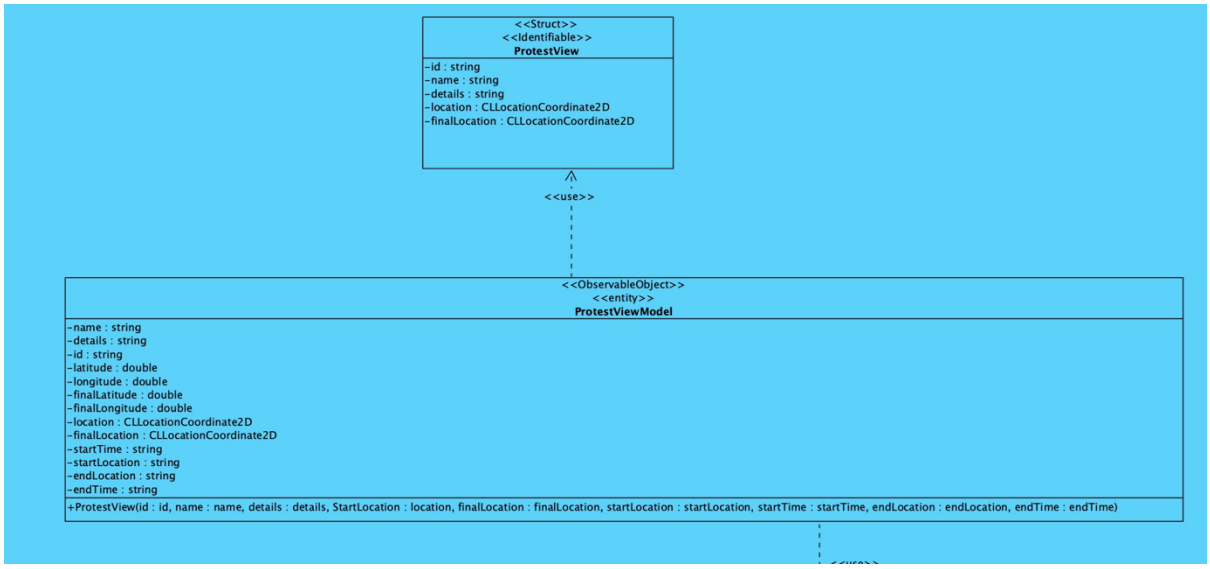
Login Class Admin



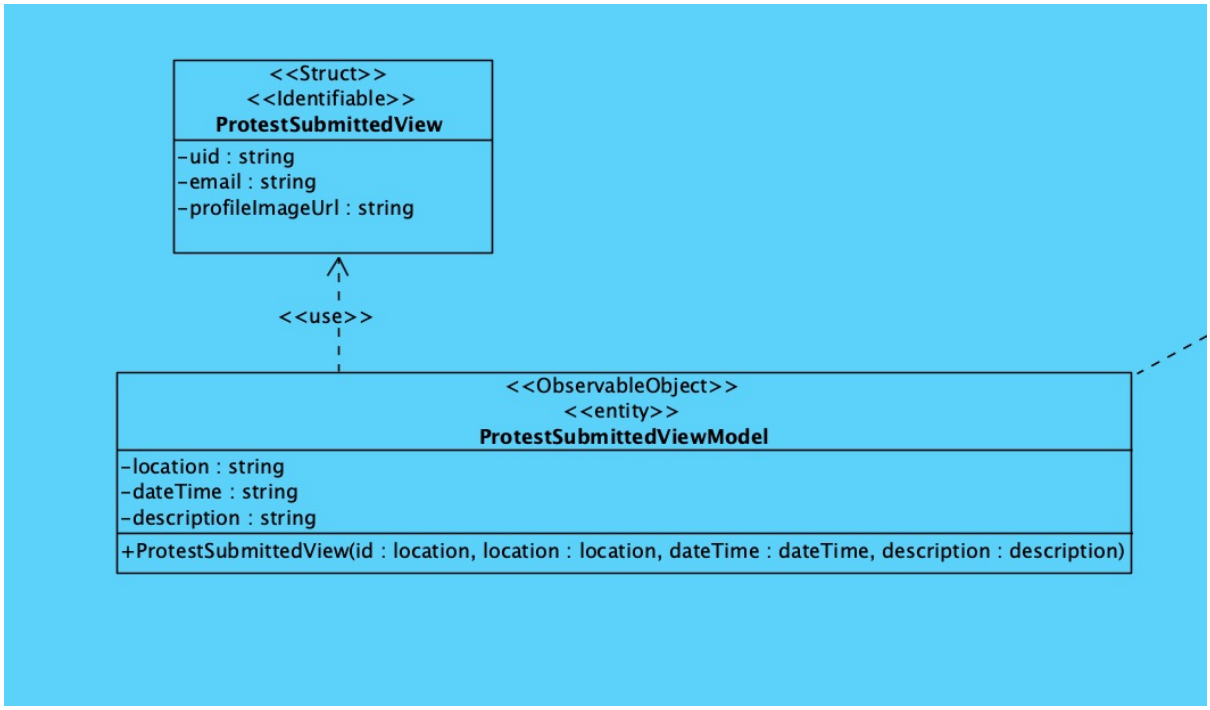
Service Database Package



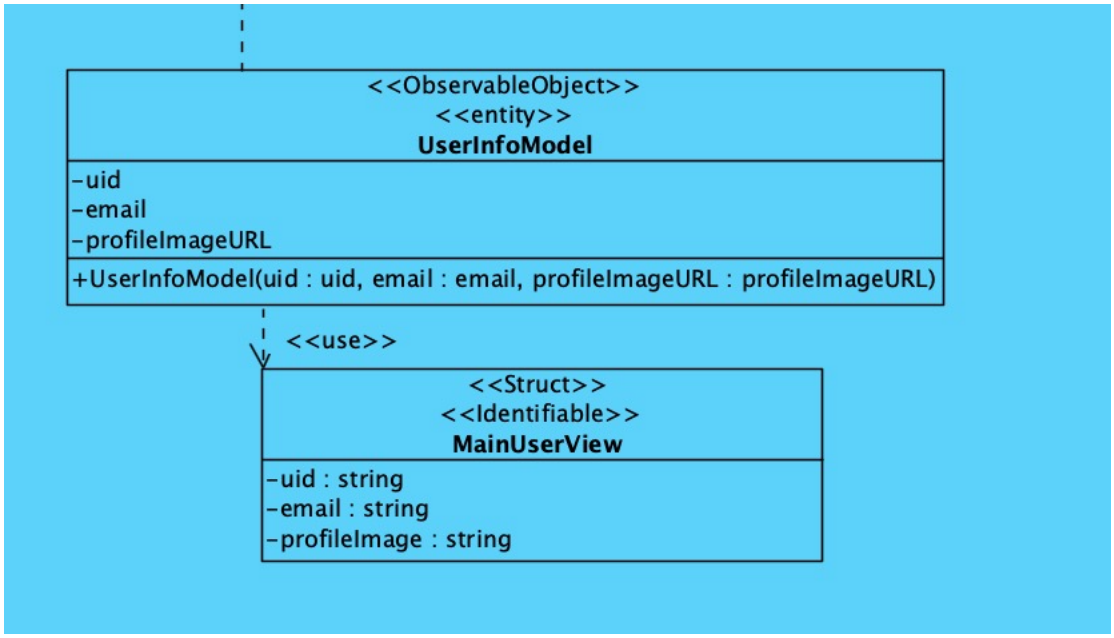
Protest View Class



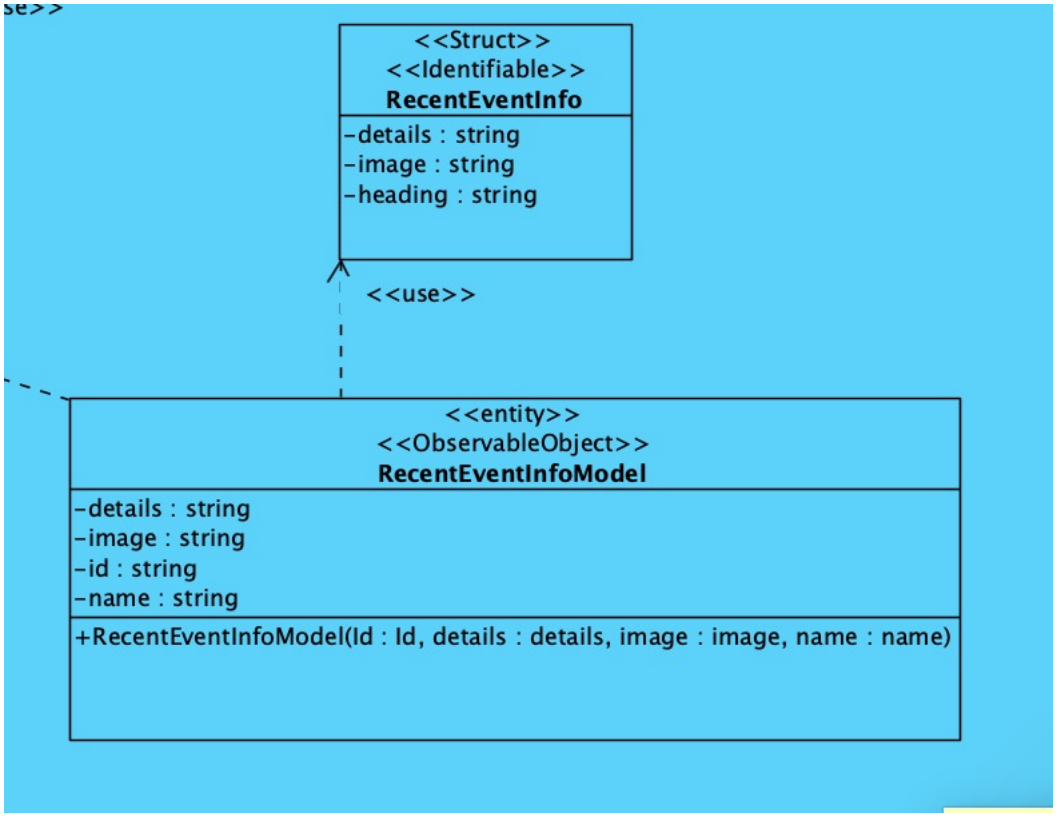
Protest Submitted Class



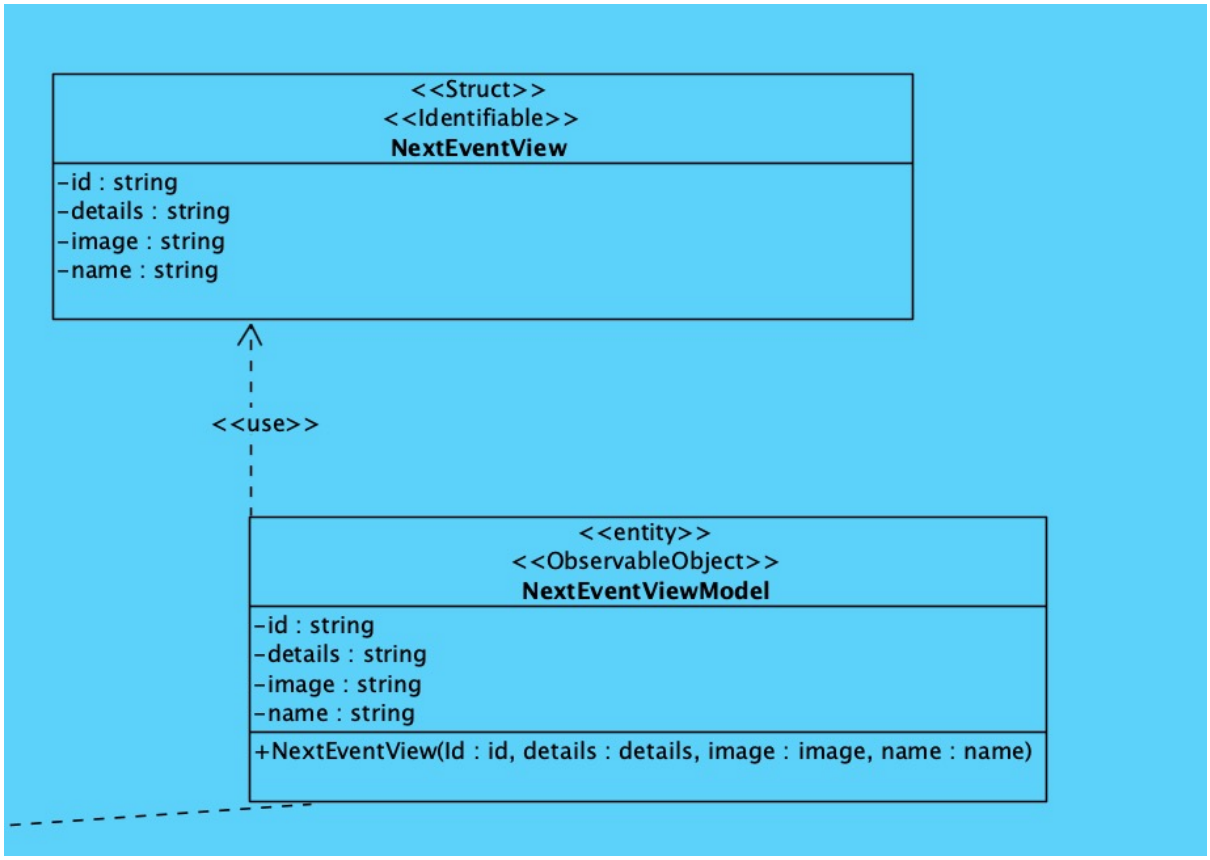
User Info Class



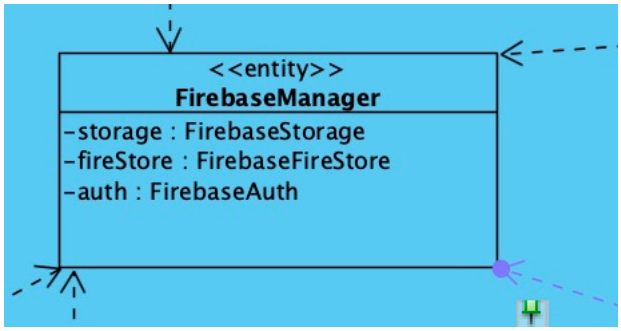
Recent Event Info Class



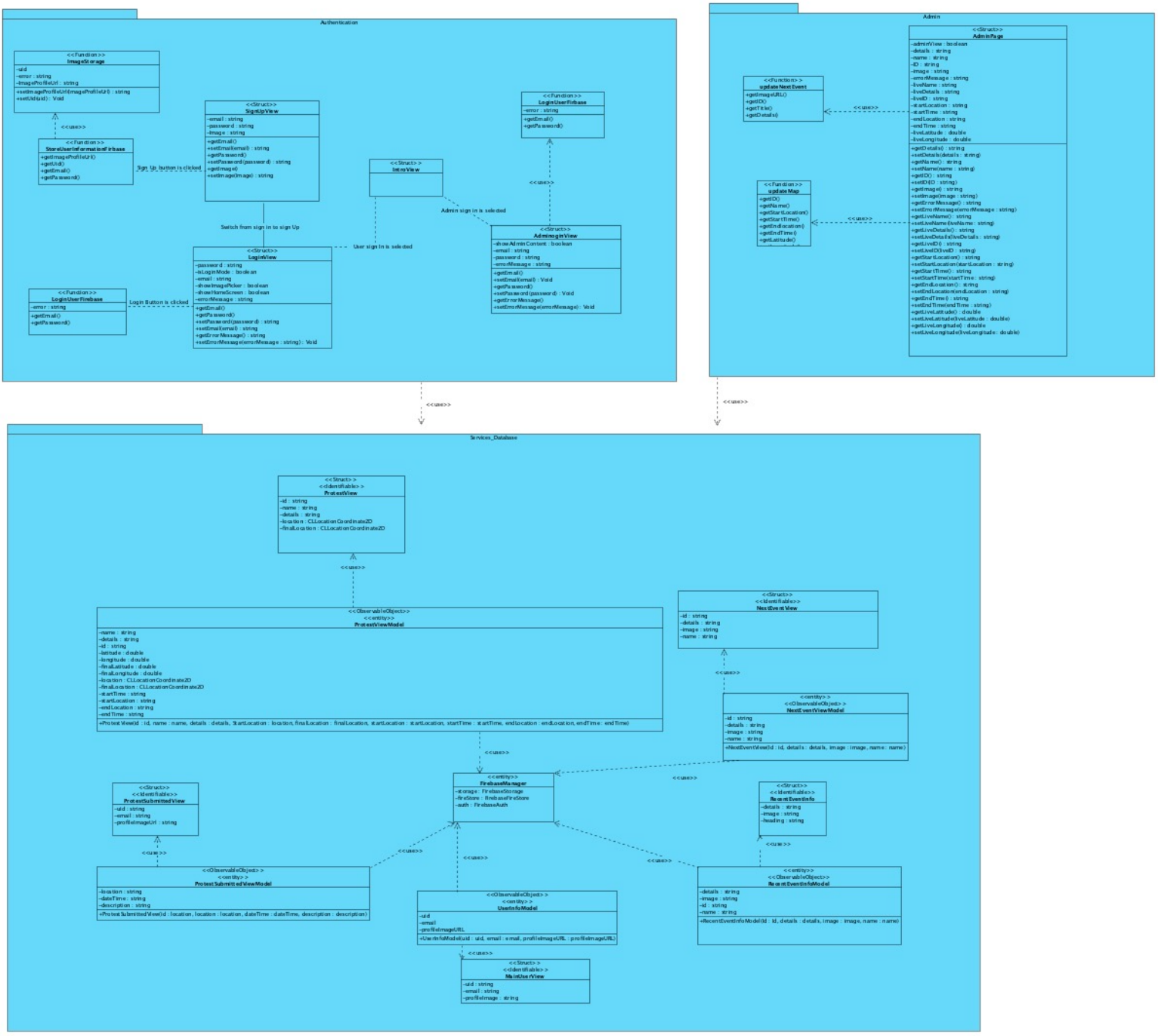
Next Event Class



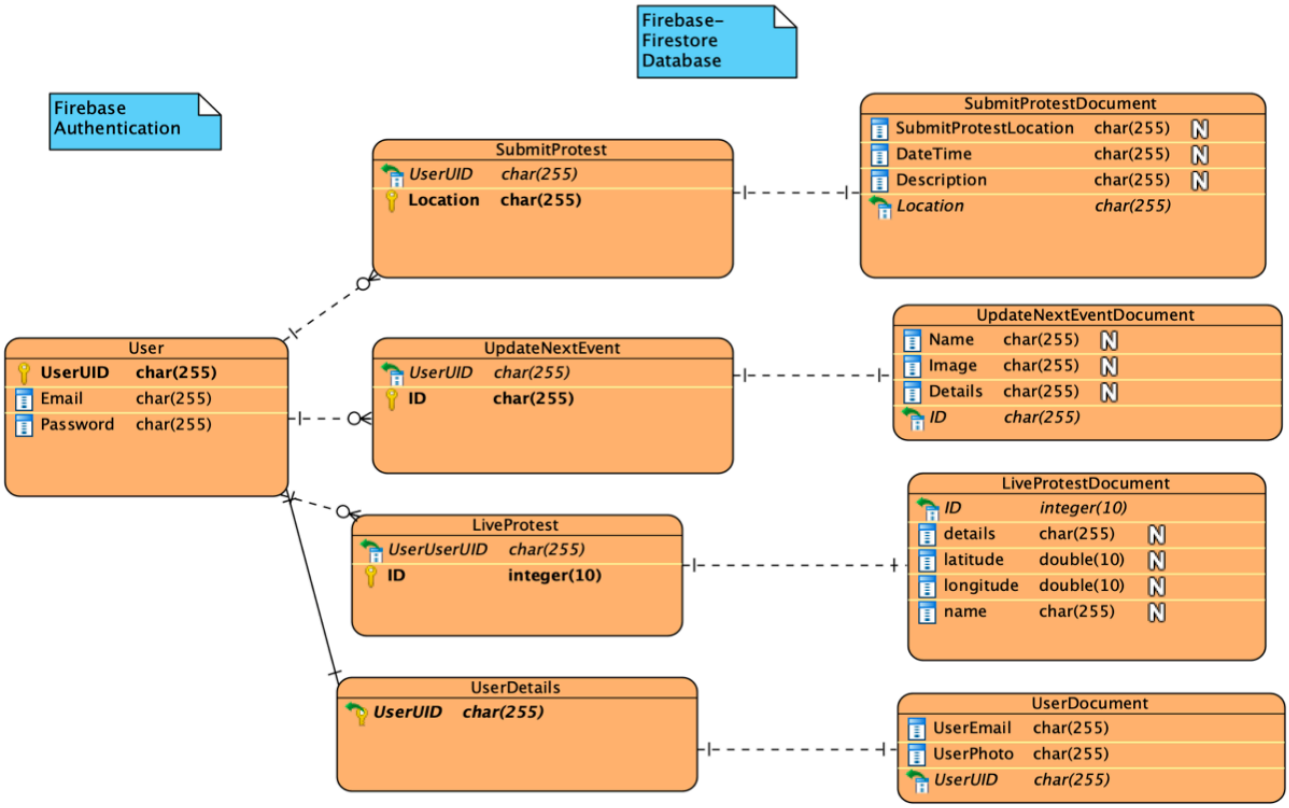
Firebase Manager Class



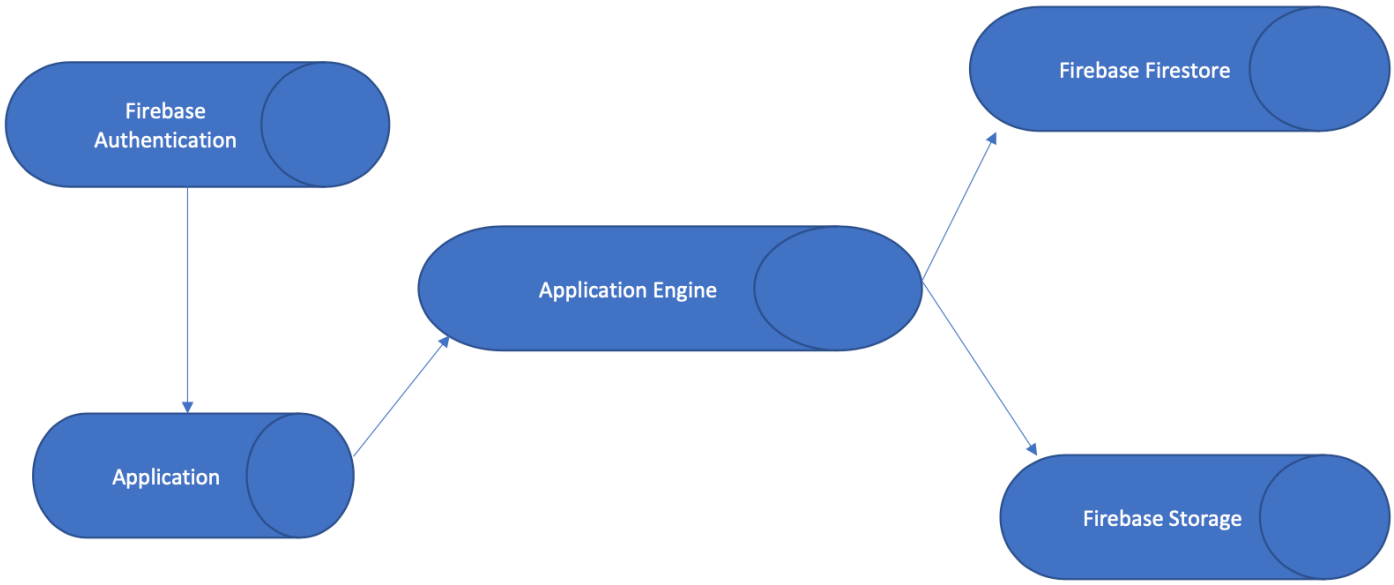
Whole Class Diagram



Entity Relationship Diagram



Cloud Architecture



APPENDIX H: IMPLEMENTATION

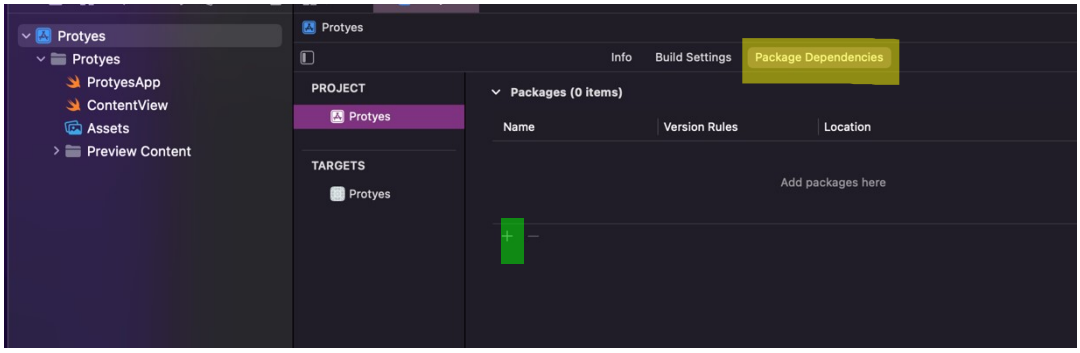
Setting up the application process

Step 1: Open the zip folder up.

Step 2: Installing Package Dependencies

Skip Step 2 if Package dependencies are already installed.

1. Navigate to the 'Package dependencies' section and click the '+' button as highlighted in the picture below



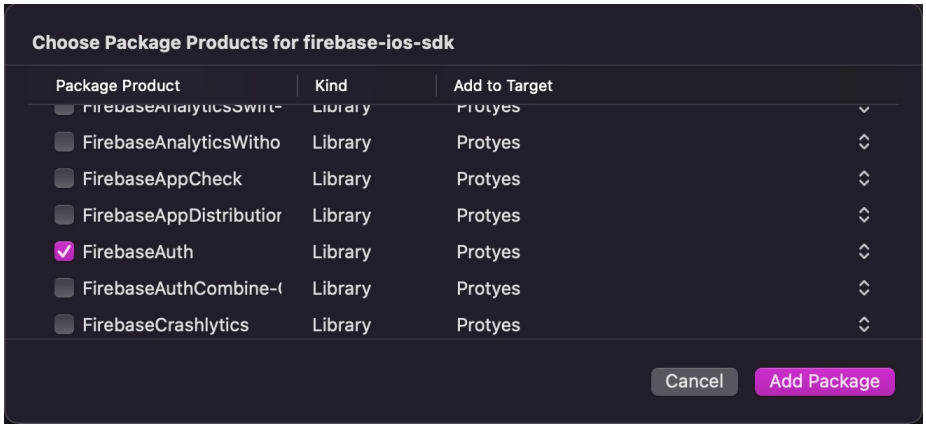
a) Install the dependencies listed below by copying the link to the package dependencies search bar (top right):

Notice – Make sure to choose 'Up to Next Major Version' on the drop box for Dependency Rules.

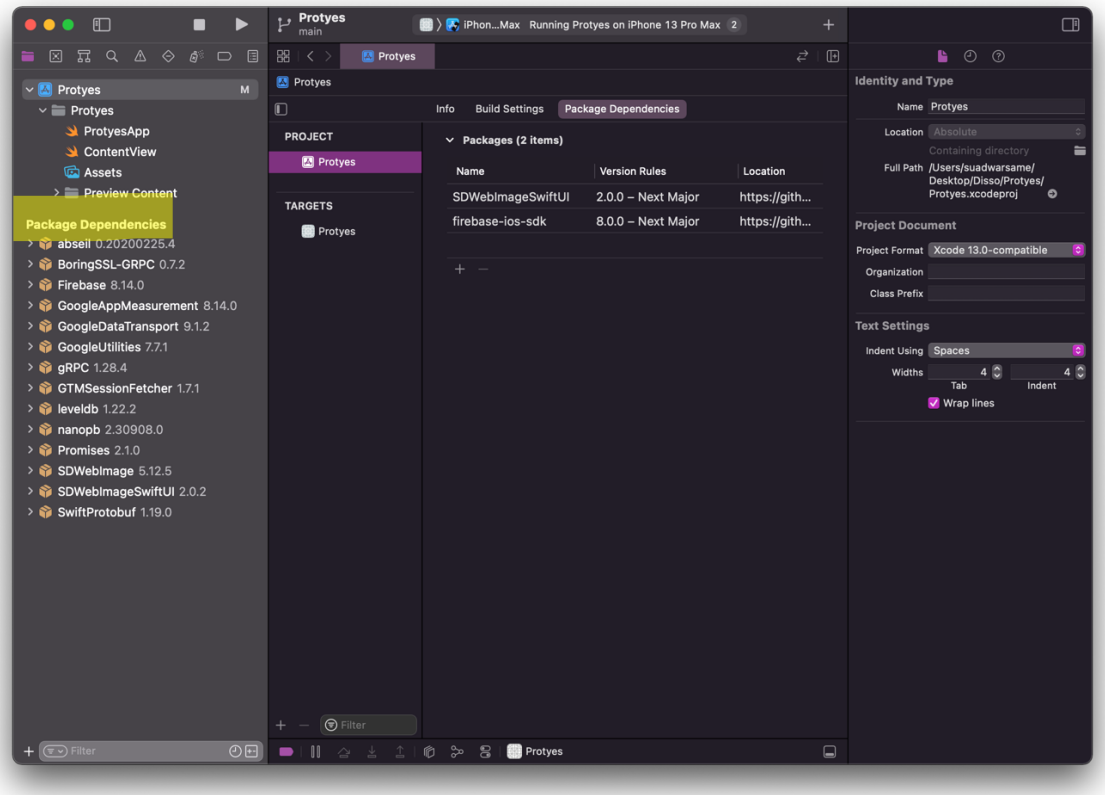
1. https://github.com/SDWebImage/SDWebImageSwiftUI
2. https://github.com/firebase/firebase-ios-sdk

b) Upon installing the second link onto the dependencies search bar, you will receive a notification similar to the picture shown below. You will need to check off these three package products then click on add package:

1. FirebaseAuth
2. FirebaseStorage
3. FirebaseFirestore

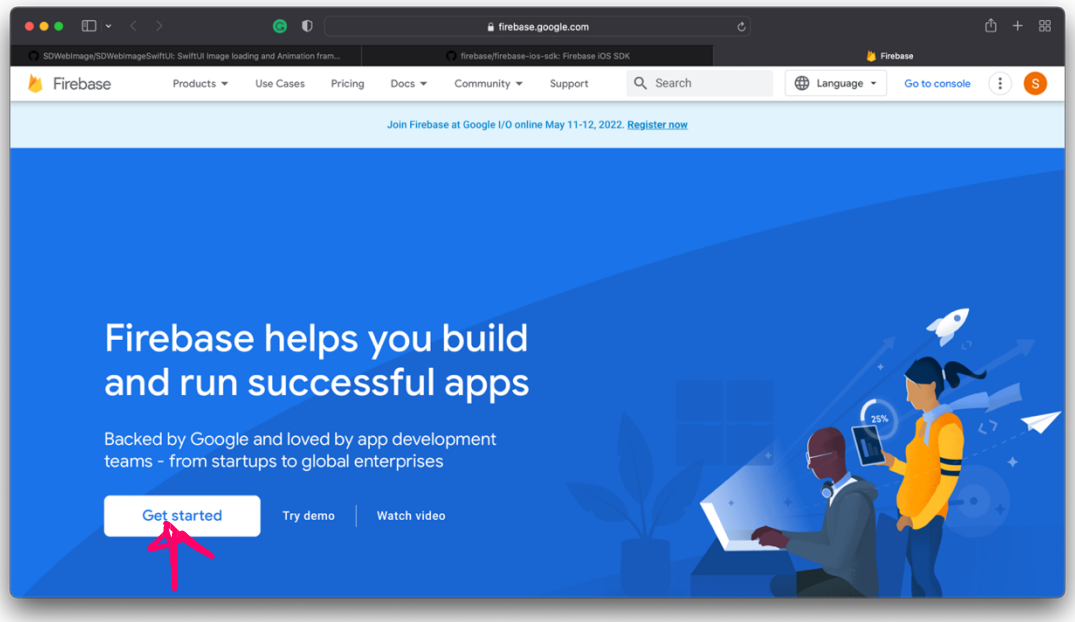


c) Once that's done, you should have these package dependencies on theside as shown below

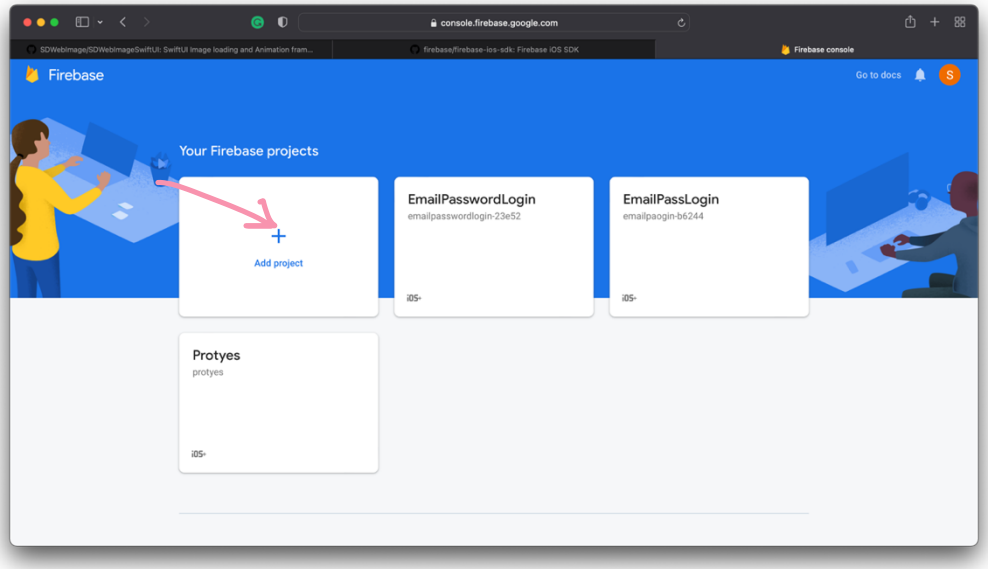


Step 3: Setting up firebase database

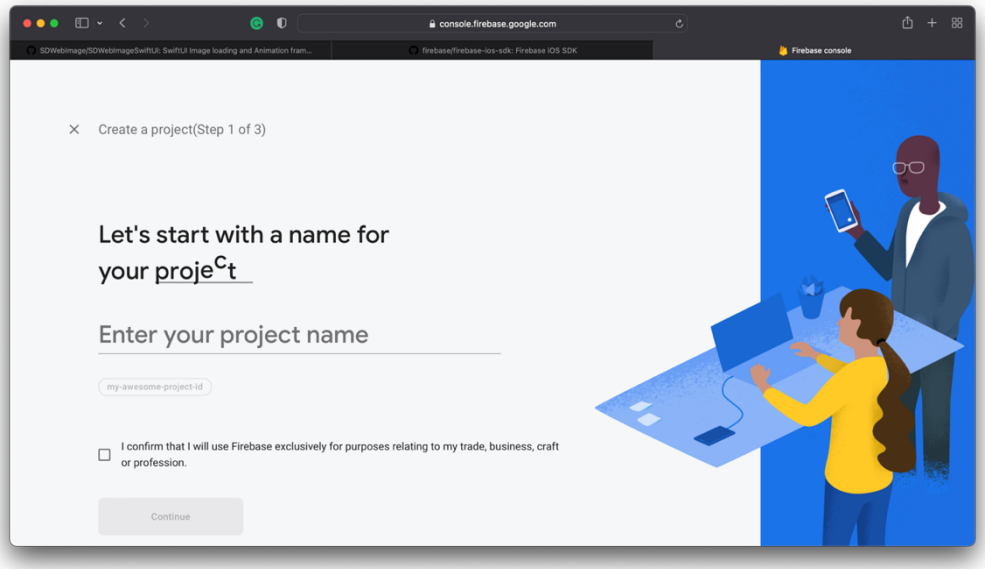
1. Type Firebase in your web browser and click on the 'Get started' button (make sure you are signed in your Google account)



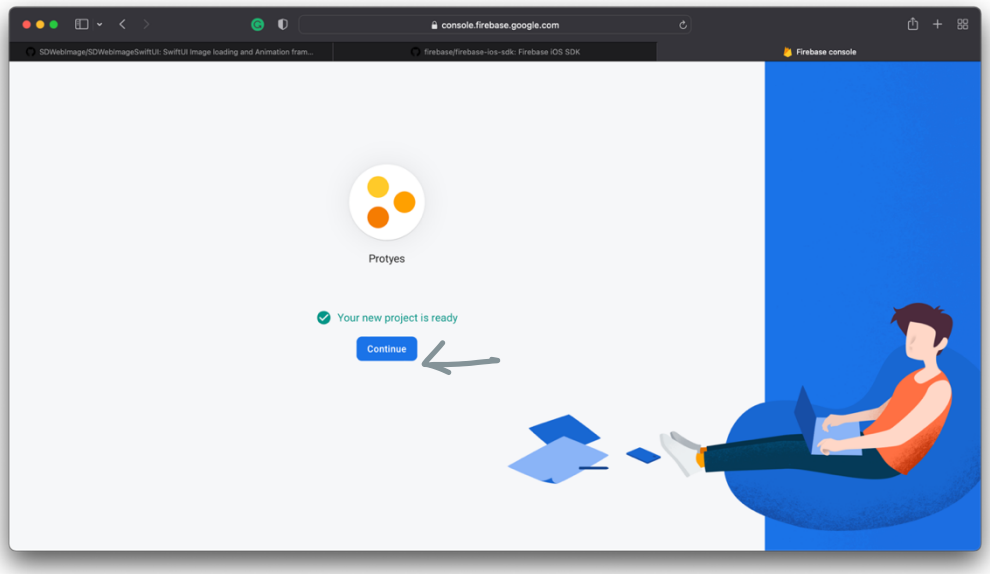
2. Click on the ‘Add project’ tab underneath ‘Your Firebase projects’3.



3. Enter a project name

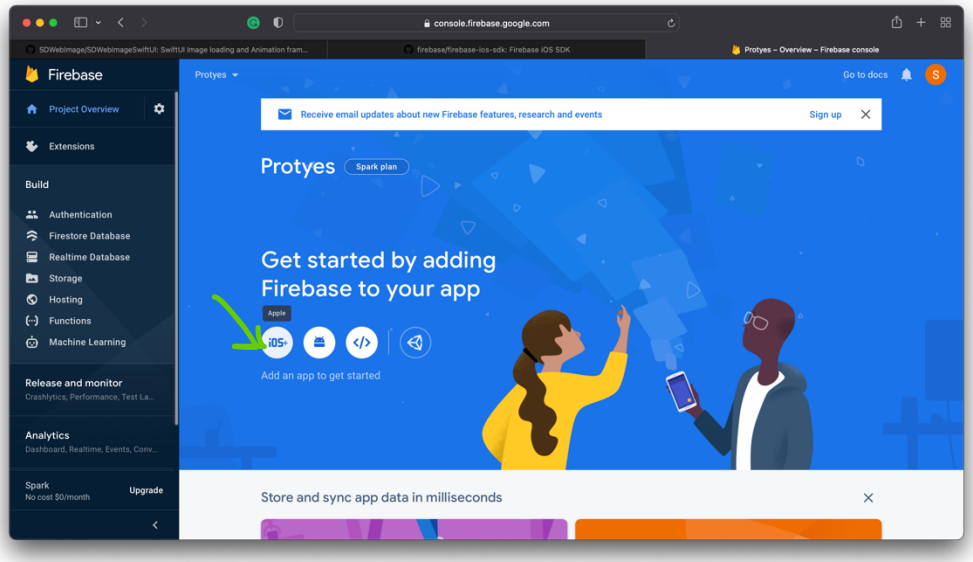


4. Project is created in database



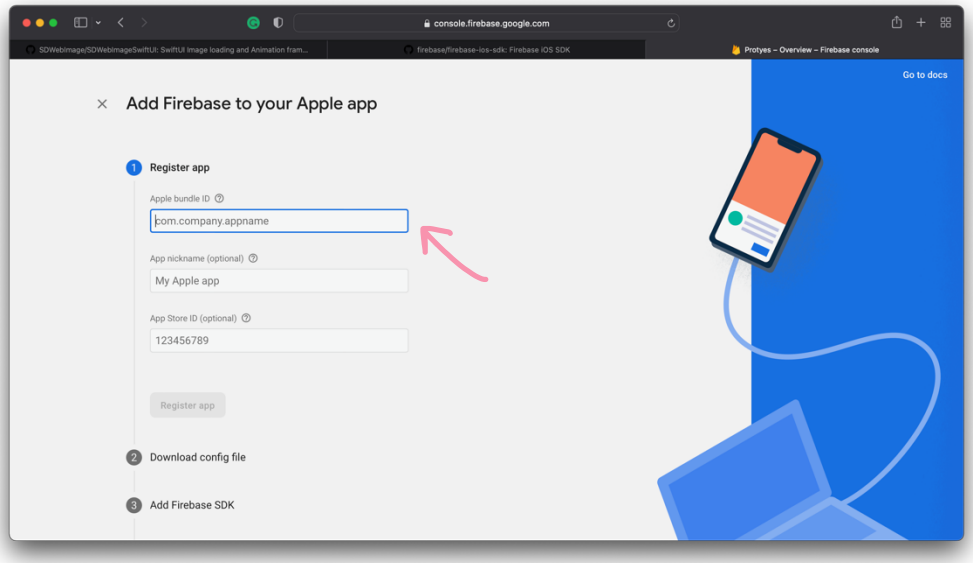
Step 4: Binding the database to XCode

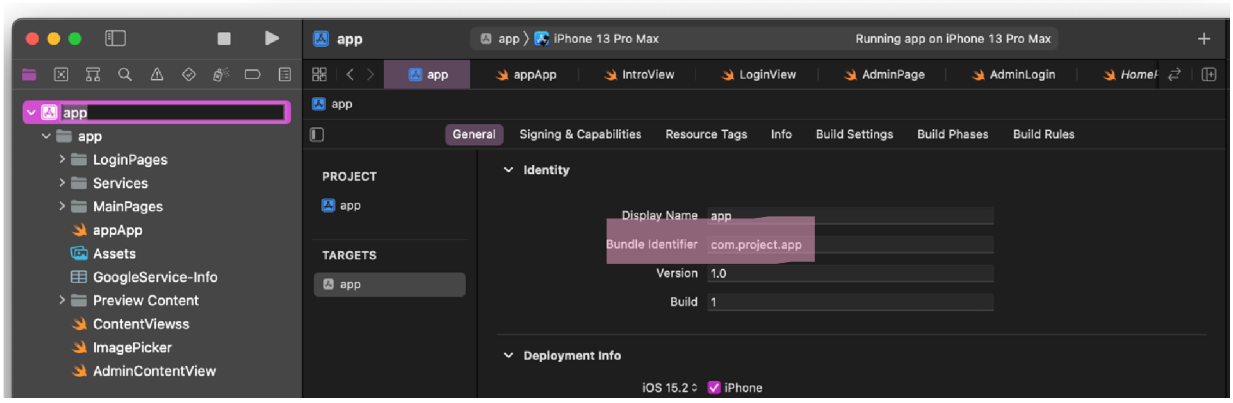
1. Click on the IOS button



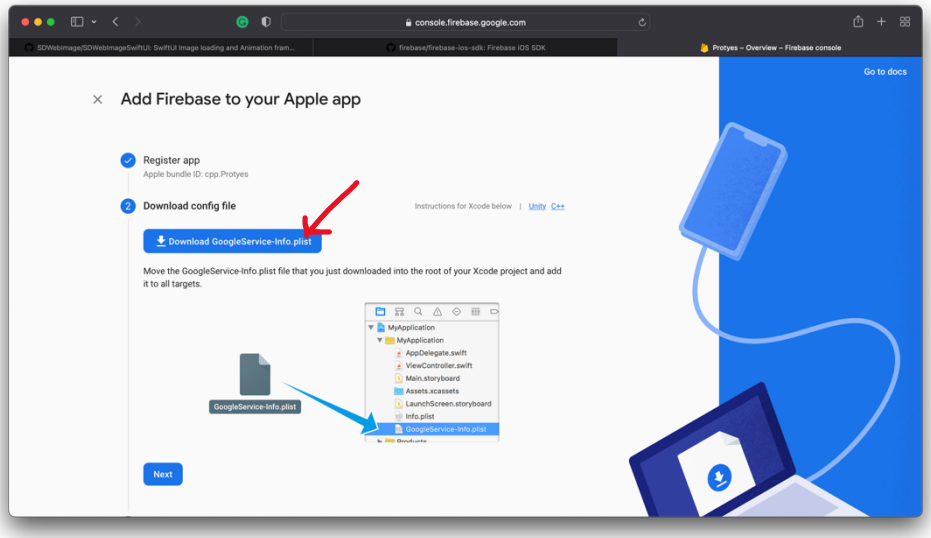
2. Copy the Apple bundle ID from XCode 'Bundle Identifier'

Other two fields are optional and can be skipped

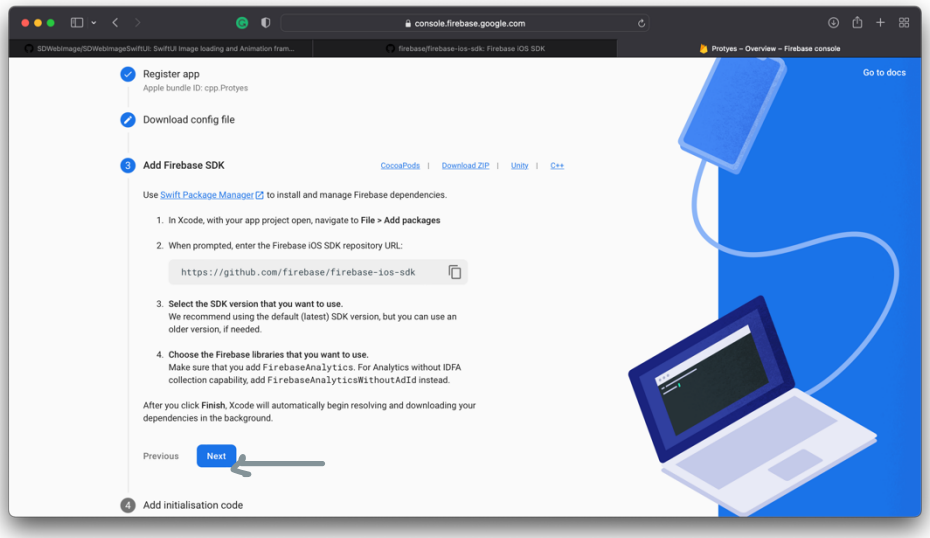




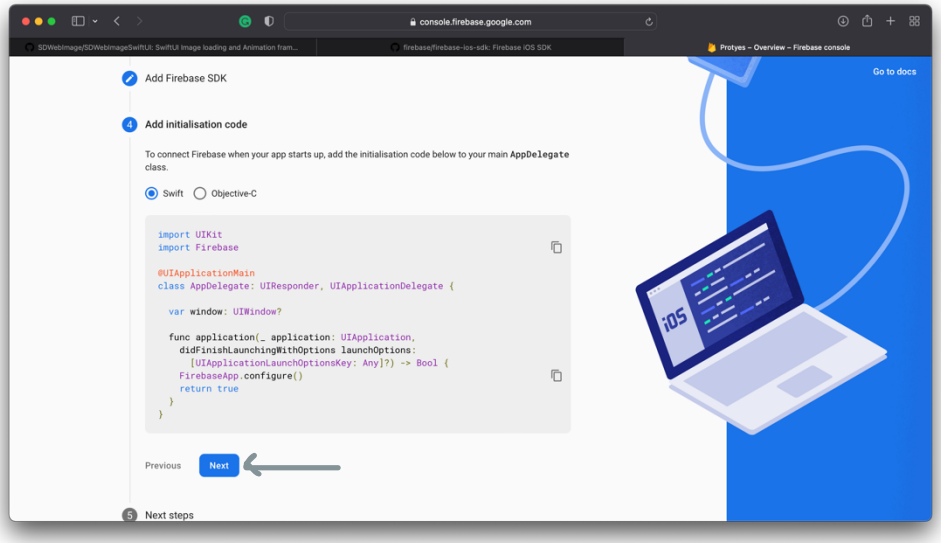
3. Download the configuration file by clicking on the button



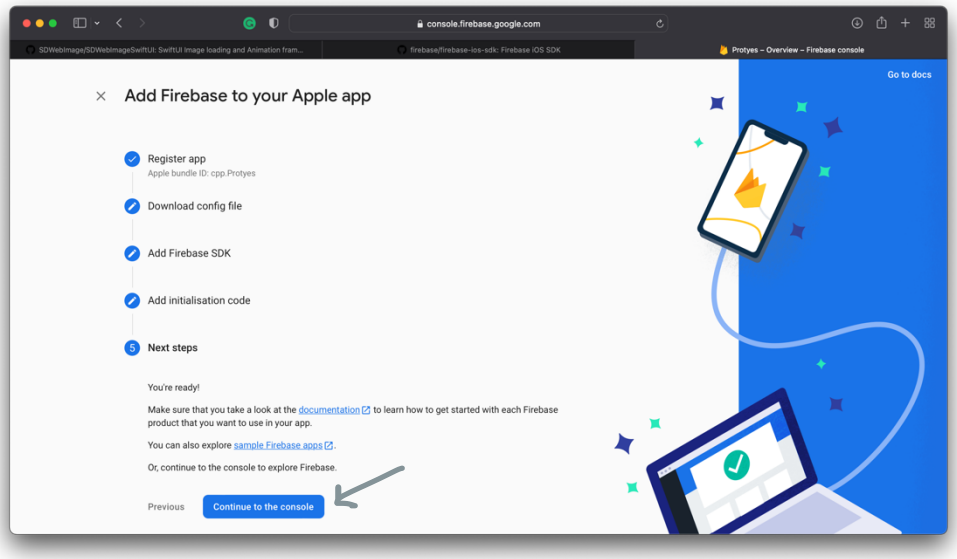
4. This step can be skipped as I already have this code implemented into the given file



5. This step can be continued as I already have this code implemented into the given file



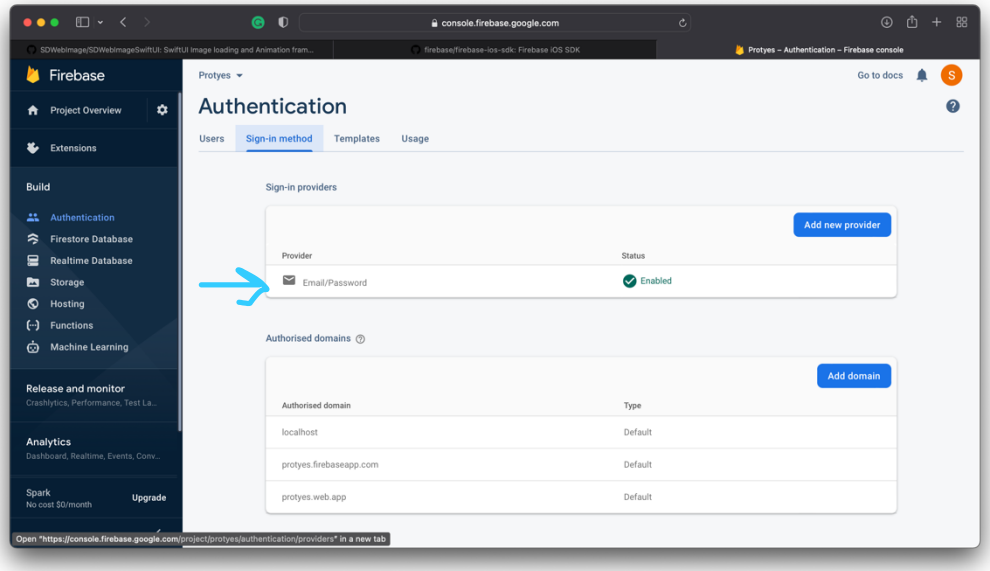
6. You have now set the firebase database to the code.



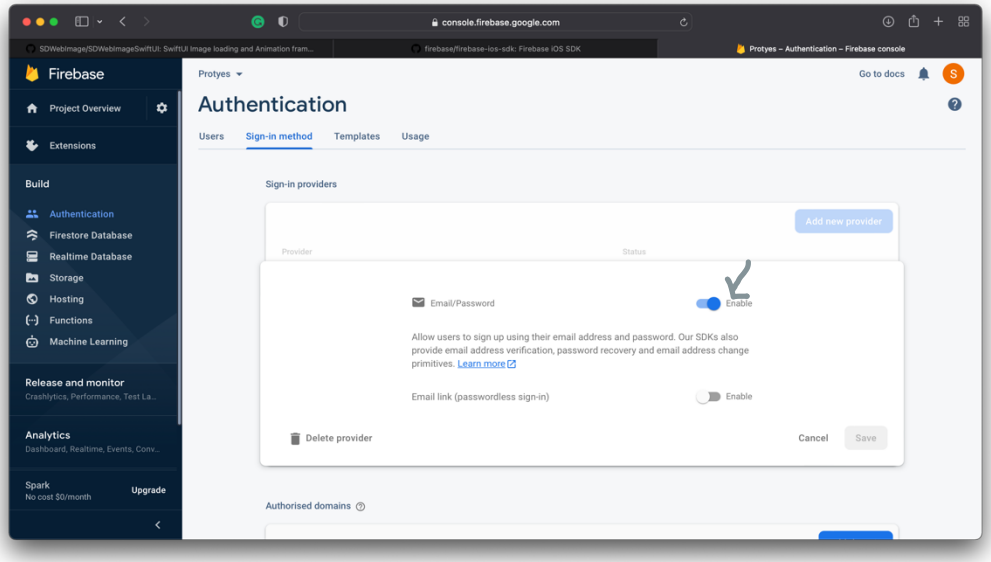
Step 5: Final checks

1. Go onto the 'Authentication' tab listed on the left

   Make sure that 'EmailPassword' is enabled

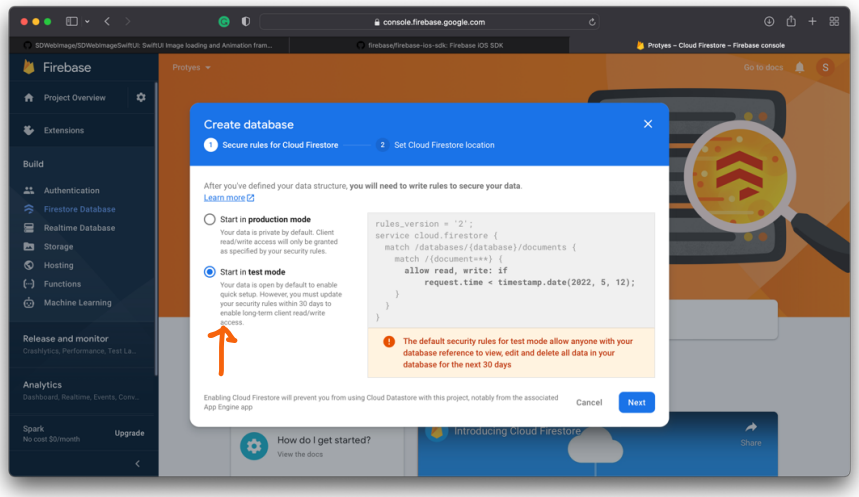


   If not follow this image and enable it, then save your changes

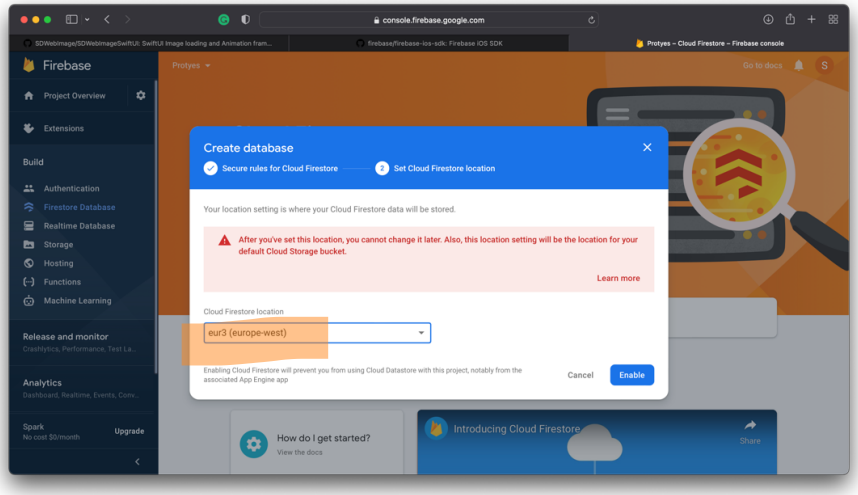


2. Navigate to the ‘Firebase Database’ tab listed on the left and click on ‘Create database’

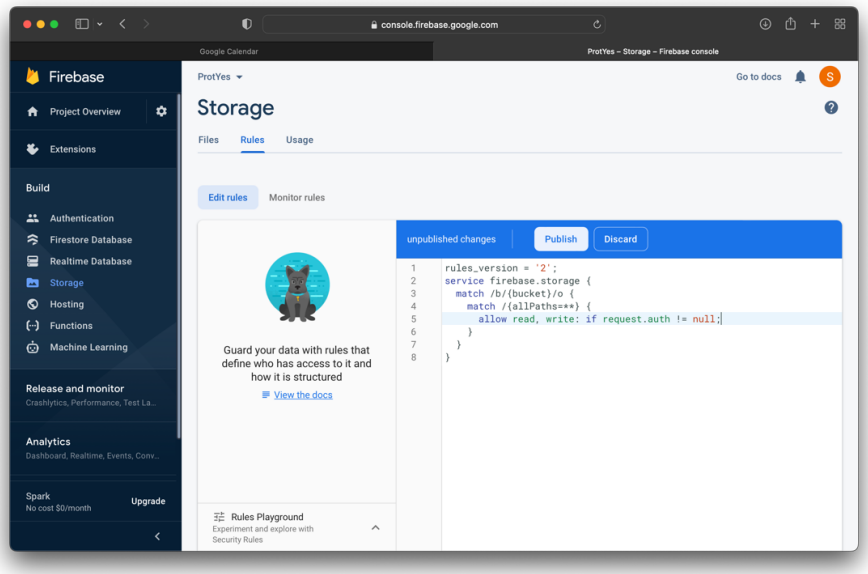
   You will see an image like this appear where you have two options, click on ‘Start in test mode’



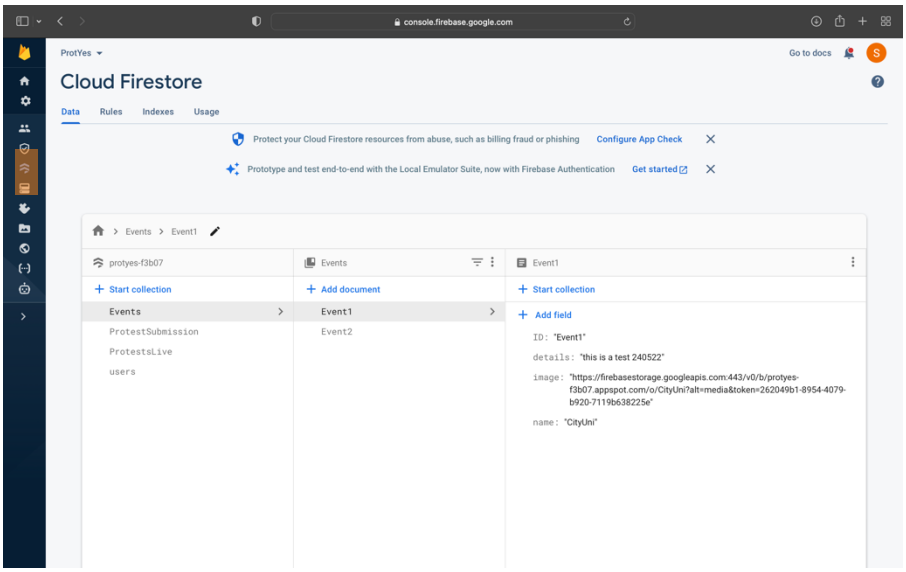
3. Change the location to ‘eur3 (europe west)’



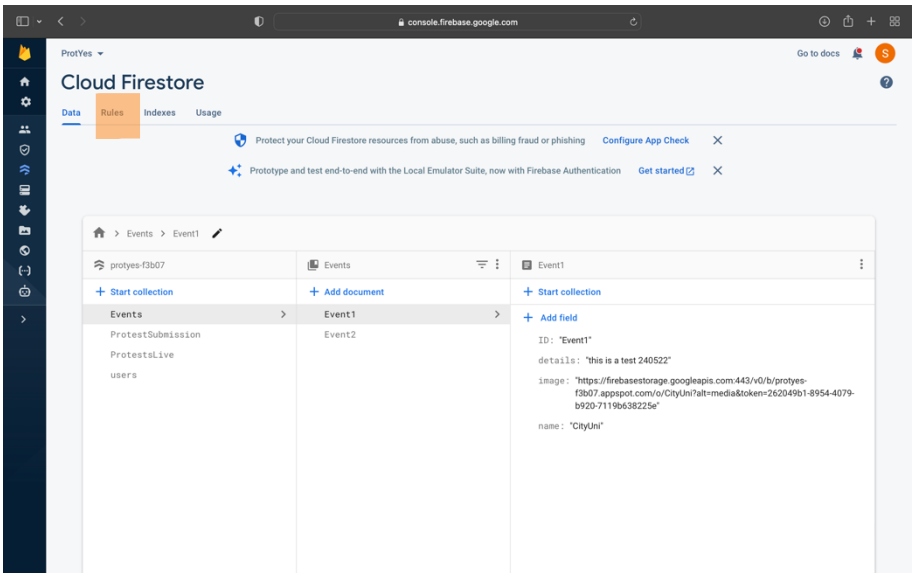
4. Finally, click on the ‘Storage’ and make sure the code in the rules is the same as the image



5.Go to ‘Firebase Firestore’ tab listed on the left



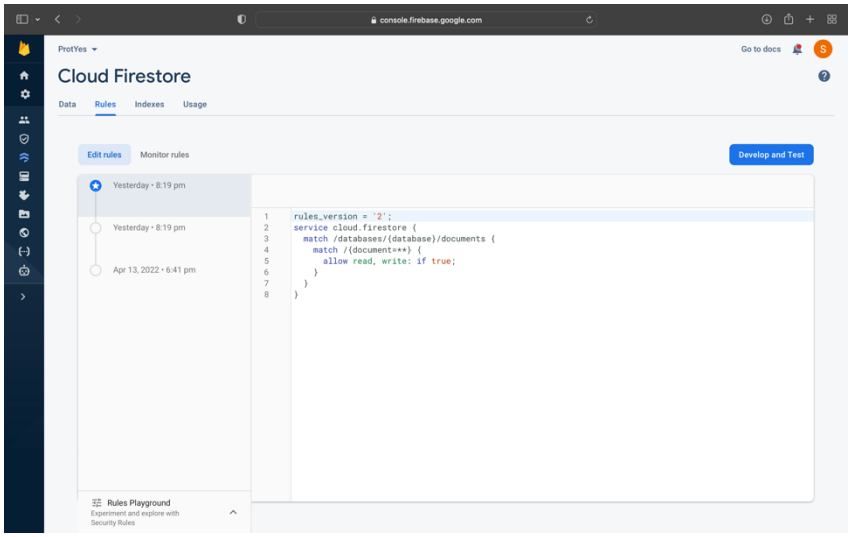
Click on the rules tab



Make sure the rule is:

```
rules_version = '2';
service cloud.firestore {
  match
/databases/{database}/documents
{
    match /{document=**} {
      allow read, write: if true;
    }
  }
}
```

As shown in the picture →



APPENDIX I: CODE WRITTEN BY AUTHOR

IntroView

```
import SwiftUI

struct IntroView: View {


    var body: some View {
        NavigationView{
            ZStack{
                Color.blue
                    .ignoresSafeArea()

                Circle()
                    .scale(1.7)
                    .foregroundColor(.white)

                VStack{
                    Text("Protyes")
                        .font(.largeTitle)
                        .bold()
                        .padding()
                        .foregroundColor(.blue)

                    Spacer()
                        .frame(height: 50)


                    NavigationLink(destination: LoginView()) {
                        Text("GET STARTED")
                            .foregroundColor(.white)
                            .frame(width: 300, height: 50)
                            .background(Color.blue)
                            .cornerRadius(30)

                }
```

```
                Spacer()
                    .frame( height: 150)

                HStack{
                    Spacer()
                        .frame(width: 200)

                    NavigationLink(destination: AdminLogin()) {
                        Text("Admin")
                            .foregroundColor(.white)
                            .frame(width: 100, height: 25)
                            .background(Color.red)
                            .cornerRadius(30)

                    }

                }

                }

            }
        }
        .navigationBarHidden(true)


    }
}

struct IntroView_Previews: PreviewProvider {
    static var previews: some View {
        IntroView()
    }
}
```

Login View

```
import Foundation
import SwiftUI
import Firebase
import FirebaseFirestore


struct LoginView: View {

    @State var isLoginMode = false
    @State var email = ""
    @State var password = ""
    @State var shouldShowImagePicker = false
    @State var showHomeScreen: Bool = false
    @State var showAdminScreen: Bool = false


    var body: some View {
        NavigationView{
            ScrollView{


                VStack(spacing: 16){
                    Picker(selection: $isLoginMode, label: Text("Picker
here"))  {
                        Text("Login")
                            .tag(true)
                        Text("Create Account")
                            .tag(false)
                    }.pickerStyle(SegmentedPickerStyle())
                        .padding()
                    if !isLoginMode{
                        Button {
                            shouldShowImagePicker.toggle()

                        } label: {
                            VStack{
```

```
            if let image = self.image {
                Image(uiImage: image)
                    .resizable()
                    .scaledToFill()
                    .frame(width: 128
                        , height: 128)
                    .cornerRadius(64)
            } else {
                Image(systemName: "person.fill")
                    .font(.system(size :64))
                    .padding()
                    .foregroundColor(Color(.label))
            }
        }
        .overlay(RoundedRectangle(cornerRadius: 64)
                .stroke(Color.black, lineWidth: 3))

    }
}
Group{
    TextField("Email", text: $email)
        .keyboardType(.emailAddress)
        .autocapitalization(.none)

    SecureField("Password", text: $password)


}.padding(12)
    .background(.white)


Button {
    handleAction()

} label: {
    HStack {
        Spacer()
```

```
                    Text(isLoginMode ? "Log In" : "Create Account")
                        .foregroundColor(.white)
                        .padding(.vertical, 10)
                        .font(.system( size: 14, weight: .semibold))
                    Spacer()
                }.background(Color.blue)


            }
            .fullScreenCover(isPresented: $showHomeScreen,
content: {
                    UserContentView()
                })
            .fullScreenCover(isPresented: $showAdminScreen,
content: {
                    AdminContentView()
                })

            Text(self.errorMessage)
                .foregroundColor(.red)
        }.padding()
      }
      .navigationTitle(isLoginMode ? "Log In" : "Create Account")
      .background(Color(.init(white: 0, alpha: 0.05))
                  .ignoresSafeArea())
    }
    .navigationViewStyle(StackNavigationViewStyle())
    .fullScreenCover(isPresented: $shouldShowImagePicker,
onDismiss: nil) {
      ImagePicker(image: $image)
    }
  }

  @State var image: UIImage?

  // Changes depending on if user is in login mode or sign up mode
  private func handleAction() {
```

```
        if isLoginMode{
            loginUser()
            //Check login deets

            //let user =
FirebaseManager.shared.auth.currentUser?.email?.contains("Admin")
         //  let Admin = "Admin"

                //Check if Email is given back

        }else {
            createNewAccount()

        }
    }

    @State var errorMessage = ""
        // Function used to create the account by using firebase Auth
        private func createNewAccount() {
            FirebaseManager.shared.auth.createUser(withEmail: email,
password: password) { result, err in
                if let err = err{
                    print("Failed to create user:", err)
                    self.errorMessage = "Failed to create user: \(err)"
                    return
                }
                print("Successfully created user : \(result?.user.uid ?? "")")
                self.errorMessage = "Successfuly created user:
\(result?.user.uid ?? "")"

                self.persistImageToStorage()

        }
    }
    // Function used to store user image URL and information to
firestore
```

```
    private func storeUserInformation(imageProfileUrl: URL) {
        guard let uid = FirebaseManager.shared.auth.currentUser?.uid
else { return }
        let userData = ["email": self.email, "uid": uid, "profileImageUrl":
imageProfileUrl.absoluteString]
        FirebaseManager.shared.firestore.collection("users")
            .document(uid).setData(userData) { err in
                if let err = err {
                    print(err)
                    self.errorMessage = "\(err)"
                    return
                }

                print("Success")
            }
    }

    //Save image to Fire Storage

    private func persistImageToStorage() {
    //    let filename = UUID().uuidString
        //Current user UID
        guard let uid = FirebaseManager.shared.auth.currentUser?.uid
        else{
            return
        }
        let ref = FirebaseManager.shared.storage.reference(withPath:
uid)
        guard let imageData =
self.image?.jpegData(compressionQuality: 0.5) else {return }
        ref.putData(imageData, metadata: nil) { metadata, err in
            if let err = err {
                self.errorMessage = "Failed to push image to Storage:
\(err)"
                return
            }
            ref.downloadURL { url, err in
```

```
            if let err = err {
                self.errorMessage = "Failed to retrieve download URL:
\(err)"
                return
            }
            self.errorMessage = "Successfully stored image with url
\(url?.absoluteString ?? "")"

            guard let url = url else {return }
            self.storeUserInformation(imageProfileUrl: url)
        }
    }

}

// Auth user by sending the inputed information to firebase auth
private func loginUser() {


    FirebaseManager.shared.auth.signIn(withEmail: email,
password: password) { result, err in
        if let err = err{
            print("Failed to login user:", err)
            self.errorMessage = "Failed to login user: \(err)"
            return
        }
        print("Successfully logged in as user : \(result?.user.uid ??
"")")
        self.errorMessage = "Successfully logged in as user:
\(result?.user.uid ?? "")"

        if FirebaseManager.shared.auth.currentUser != nil {
            showHomeScreen = true

          }
    }
```

```
    }
}

struct LoginView_Previews: PreviewProvider {
    static var previews: some View {
        LoginView()
    }
}
```

AdminView

```
import SwiftUI

struct AdminLogin: View {

    @State var showAdminContent = false
    @State var password = ""
    @State var email = "admin@gmail.com"
    @State var errorMessage = ""

    var body: some View {
        NavigationView{
            ZStack{
                Color.gray
                    .ignoresSafeArea()

                Circle()
                    .scale(1.7)
                    .foregroundColor(.white)

                VStack{
                    Text("Admin")
                        .font(.largeTitle)
                        .bold()
                        .padding()
                        .foregroundColor(.blue)

                    Spacer()
                        .frame(height: 50)
                    SecureField("Password", text: $password)
                        .padding()
                        .frame(width: 300, height: 50)
                        .background(Color.black.opacity(0.05))
                        .cornerRadius(10)
                    Spacer()
                        .frame(height: 50)
                    Button {
```

```
                loginUser()
            } label: {
                HStack {
                    Spacer()
                    Text("Sign In")
                        .foregroundColor(.white)
                        .padding(.vertical, 10)
                        .font(.system( size: 14, weight: .semibold))
                        .frame(width: 300, height: 50)
                    Spacer()
                }.background(Color.blue)
            }
            .fullScreenCover(isPresented: $showAdminContent,
content: {

                AdminContentView()
            })

            Text(self.errorMessage)
                .foregroundColor(.red)

            }

        }
    }
    .navigationBarHidden(false)
}

// Directs admin to admin page if the correct password is inputted
private func handleAction() {
    loginUser()

    if FirebaseManager.shared.auth.currentUser != nil {
        showAdminContent = true

    } else{
            showAdminContent = false
```

```
            }
        }

        // User information is sent to firebase Auth to get verfied
        private func loginUser() {
            FirebaseManager.shared.auth.signIn(withEmail: email,
password: password) { result, err in
                if let err = err{
                    print("Failed to login user:", err)
                    self.errorMessage = "Failed to login user: \(err)"
                    return
                }
                print("Successfully logged in as user : \(result?.user.uid ??
"")")
                self.errorMessage = "Successfully logged in as user:
\(result?.user.uid ?? "")"

                if FirebaseManager.shared.auth.currentUser != nil {
                    showAdminContent = true }

            }


        }
}

struct AdminLogin_Previews: PreviewProvider {
    static var previews: some View {
        AdminLogin()
    }
}
```

Firebase Manager

```
import Foundation
import Firebase

class FirebaseManager: NSObject {
    let auth: Auth
    let storage: Storage
    let firestore: Firestore

    static let shared = FirebaseManager()

    override init() {
        FirebaseApp.configure()
        self.auth = Auth.auth()
        self.storage = Storage.storage()
        self.firestore = Firestore.firestore()

        super.init()
    }
}
```

Database Model

```
import Foundation
import SwiftUI
import FirebaseFirestore
import MapKit

// Model for the user profile information

struct MainUser {
    let uid, email, profileImageUrl: String
}


class UserInfoModel: ObservableObject{


    @Published var email = ""
    @Published var errorMessage = ""
    @Published var profileImage = ""

    init() {
        fetchCurrentUser()
    }
    private func fetchCurrentUser() {
        guard let uid = FirebaseManager.shared.auth.currentUser?.uid
else{
            self.errorMessage = "No UID found"
            return
        }


FirebaseManager.shared.firestore.collection("users").document(uid).g
etDocument { snapshot, error in
            if let error = error{
                print("failed to fetch current user", error)
                return
            }
```

```
            guard let data = snapshot?.data() else{
                return
            }


            let uid = data["uid"] as? String ?? ""
            let email = data["email"] as? String ?? ""
            let profileImageUrl = data["profileImageUrl"] as? String ?? ""
            let chatUser = MainUser(uid: uid, email: email,
profileImageUrl: profileImageUrl)


            self.email = chatUser.email
            self.profileImage = chatUser.profileImageUrl
        }


    }
}


// Database model for live protest infromation that will be plotted on
the map
struct ProtestView: Identifiable {
 var id: String
 var name: String
 var details : String
    var startLocation: String
    var startTime : String
    var endLocation : String
    var endTime : String
 var cooridinate : CLLocationCoordinate2D
 var finalCoordinate: CLLocationCoordinate2D
}

//Information from firestore used for the live map
class ProtestViewModel: ObservableObject {
```

```
  @Published var cities = [ProtestView]()

    private var db = FirebaseManager.shared.firestore
    @Published var errorMessage = ""

  func fetchCities() {
      db.collection("ProtestsLive").addSnapshotListener{
(querySnapshot, error) in
          guard let documents = querySnapshot?.documents else {
              print("no documents")
              return
          }
          self.cities = documents.map { (queryDocumentSnapshot) ->
ProtestView in
              let data = queryDocumentSnapshot.data()


              let name = data["name"] as? String ?? ""
              let details = data["details"] as? String ?? ""
              let startLocation = data["startLocation"] as? String ?? ""
              let startTime = data["startTime"] as? String ?? ""
              let endLocation = data["endLocation"] as? String ?? ""
              let endTime = data["endTime"] as? String ?? ""
              let id = data["id"] as? String ?? ""
              let latitude = data["latitude"] as? Double ?? 0
             // let g = latitude
              let longitude = data["longitude"] as? Double ?? 0
              let finalLatitude = data["Finallatitude"] as? Double ?? 0
              let finalLongitude = data["Finallongitude"] as? Double ?? 0

              let location = CLLocationCoordinate2D(latitude: latitude,
longitude: longitude)
              let finalLocation = CLLocationCoordinate2D(latitude:
finalLatitude, longitude: finalLongitude)

              return ProtestView(id: id, name: name,details: details,
startLocation: startLocation,startTime: startTime, endLocation:
```

```
endLocation, endTime: endTime, cooridinate:
location,finalCoordinate: finalLocation)

        }
    }
  }
}

//Submited info data

struct ProtestSubmittedView: Identifiable {
 var id: String

 var location: String
 var dateTime: String
 var description : String
}


class ProtestSubmittedViewModel: ObservableObject {
  @Published var info = [ProtestSubmittedView]()

    private var db = FirebaseManager.shared.firestore
    @Published var errorMessage = ""

  func fetchInfo() {
      db.collection("ProtestSubmission").addSnapshotListener{
(querySnapshot, error) in
          guard let documents = querySnapshot?.documents else {
              print("no documents")
              return
          }
          self.info = documents.map { (queryDocumentSnapshot) ->
ProtestSubmittedView in
              let data = queryDocumentSnapshot.data()
```

```
            let location = data["location"] as? String ?? ""
            let dateTime = data["dateTime"] as? String ?? ""
            let description = data["description"] as? String ?? ""


            return ProtestSubmittedView(id: location, location: location,
dateTime: dateTime, description: description)

        }


    }



  }
}


struct NextEventInfo: Identifiable {
 var id: String

 var details: String
 var image: String
 var name : String
}

class NextEventViewModel: ObservableObject {
  @Published var info = [NextEventInfo]()

    private var db = FirebaseManager.shared.firestore
    @Published var errorMessage = ""

  func fetchInfo() {
      db.collection("Events").addSnapshotListener{ (querySnapshot,
error) in
          guard let documents = querySnapshot?.documents else {
```

```
            print("no documents")
            return
        }
        self.info = documents.map { (queryDocumentSnapshot) ->
NextEventInfo in
            let data = queryDocumentSnapshot.data()


            let ID = data["ID"] as? String ?? ""
            let details = data["details"] as? String ?? ""
            let image = data["image"] as? String ?? ""
            let name = data["name"] as? String ?? ""



            return NextEventInfo(id: ID, details: details, image: image,
name: name)



        }


    }



  }
}

//Next event data from firebase


struct RecentEventInfo {
   let ID, details, image, name1: String
}
```

```
class RecentProtestInfo: ObservableObject{


    @Published var details = ""
    @Published var image1 = ""
    @Published var heading = ""

    init() {
        recentPosts()
    }
    private func recentPosts() {



FirebaseManager.shared.firestore.collection("Events").document("Ev
ent1").getDocument { snapshot, error in
            if let error = error{
                print("failed to fetch document", error)
                return
            }
            guard let data = snapshot?.data() else{
                return
            }


            let ID = data["ID"] as? String ?? ""
            let details = data["details"] as? String ?? ""
            let image = data["image"] as? String ?? ""
            let name1 = data["name"] as? String ?? ""
            let recentPostsInfo = RecentEventInfo(ID: ID, details: details,
image: image, name1: name1)


            self.heading = recentPostsInfo.name1
            self.details = recentPostsInfo.details
            self.image1 = recentPostsInfo.image
```

```
        }
    }
}

class RecentProtestInfo23: ObservableObject{


    @Published var details = ""
    @Published var image1 = ""
    @Published var heading = ""

    init() {
        recentPosts()
    }
    private func recentPosts() {


FirebaseManager.shared.firestore.collection("Events").document("Ev
ent1").getDocument { (document, error) in
            if let document = document, document.exists {
                let dataDescription =
document.data().map(String.init(describing:)) ?? "nil"
                print("Document data: \(dataDescription)")

                let data = document.data()
                let ID = data?["ID"] as? String ?? ""
                let details = data?["details"] as? String ?? ""
                let image = data?["image"] as? String ?? ""
                let name1 = data?["name"] as? String ?? ""
                let recentPostsInfo = RecentEventInfo(ID: ID, details:
details, image: image, name1: name1)


                self.heading = recentPostsInfo.name1
                self.details = recentPostsInfo.details
                self.image1 = recentPostsInfo.image
```

```
            } else {
                print("Document does not exist")
            }
        }

    }
}



class RecentProtestInfo2: ObservableObject{

    @Published var details = ""
    @Published var image1 = ""
    @Published var heading = ""

    init() {
        recentPosts()
    }
    private func recentPosts() {



FirebaseManager.shared.firestore.collection("Events").document("Ev
ent2").getDocument { snapshot, error in
            if let error = error{
                print("failed to fetch document", error)
                return
            }
            guard let data = snapshot?.data() else{
                return
            }



            let ID = data["ID"] as? String ?? ""
            let details = data["details"] as? String ?? ""
```

```
            let image = data["image"] as? String ?? ""
            let name1 = data["name"] as? String ?? ""
            let recentPostsInfo = RecentEventInfo(ID: ID, details: details,
image: image, name1: name1)



            self.heading = recentPostsInfo.name1
            self.details = recentPostsInfo.details
            self.image1 = recentPostsInfo.image


        }


    }
}
```

HomePageView

```
import SwiftUI
import MapKit
import FirebaseFirestore

struct Place: Identifiable {
    let id = UUID()
    let name: String
    let cooridinate : CLLocationCoordinate2D

}


struct HomePageView: View {

    @State private var region = MKCoordinateRegion(center:
CLLocationCoordinate2D(latitude: 51.507351, longitude: -0.127758),
latitudinalMeters: 5000, longitudinalMeters: 5000)




    @ObservedObject private var viewModel = ProtestViewModel()
    var body: some View {

        NavigationView{
            Map(coordinateRegion: $region, annotationItems:
viewModel.cities) {
                cities in



                MapAnnotation(coordinate: cities.cooridinate ){

                    NavigationLink{
```

```
                ZStack{
                    Color.red


                    VStack{
                        Text("WARNING!!!!")
                            .padding()
                            .background(Color(hex: "ffff00"))
                            .foregroundColor(.black)
                            .font(.system(size: 45, weight: .bold, design:
.default))

                        Text("-----Live Protest-----")
                            .font(.largeTitle)
                            .bold()
                            .padding()
                            .foregroundColor(.black)
                        Text("Start Location: " + cities.startLocation)
                            .bold()
                        Text("Start Time: " + cities.startTime)
                            .bold()
                        Text("End location: " + cities.endLocation)
                            .bold()
                        Text("EndTime: " + cities.endTime)
                            .bold()
                        Text(cities.name)
                            .bold()
                            .font(.title)
                        Spacer()
                            .frame(width: 10, height: 10)
                        Text(cities.details)
                            .multilineTextAlignment(.center)


                        Spacer()
                            .frame(width : 100, height: 100)
```

```
                }


            }


        } label: {

            VStack{
                Text("PROTEST")
                    .foregroundColor(Color.red)

                Circle()

                    .stroke(.red, lineWidth: 5)
                    .frame(width: 47, height:47)
                    .background(Color.init(
                        red: 100, green: 0, blue: 0, opacity: 0.3))
                    .cornerRadius(10)
            }



        }
    }

    // Code that will be used to implement direction for map
    /**
    // NYC
    let p1 = MKPlacemark(coordinate:
CLLocationCoordinate2D(latitude: 40.71, longitude: -74))

    // Boston
    let p2 = MKPlacemark(coordinate:
CLLocationCoordinate2D(latitude: 42.36, longitude: -71.05))

    let request = MKDirections.Request()
```

```
            request.source = MKMapItem(placemark: p1)
            request.destination = MKMapItem(placemark: p2)
            request.transportType = .automobile

            let directions = MKDirections(request: request)
            directions.calculate { response, error in
              guard let route = response?.routes.first else { return }
              mapView.addAnnotations([p1, p2])
              mapView.addOverlay(route.polyline)
              mapView.setVisibleMapRect(
                route.polyline.boundingMapRect,
                edgePadding: UIEdgeInsets(top: 20, left: 20, bottom: 20,
right: 20),
                animated: true)
              self.directions = route.steps.map { $0.instructions }.filter {
!$0.isEmpty }
            }

            */









        }.ignoresSafeArea()
            .onAppear(){
                self.viewModel.fetchCities()
            }

    }.navigationTitle("Protest info")
  }

  // Code that will be used to impplement direction for the map
  class MapViewCoordinator: NSObject, MKMapViewDelegate {
```

```
        func mapView(_ mapView: MKMapView, rendererFor overlay:
MKOverlay) -> MKOverlayRenderer {
            let renderer = MKPolylineRenderer(overlay: overlay)
            renderer.strokeColor = .systemBlue
            renderer.lineWidth = 5
            return renderer
        }
    }
}

struct HomePageView_Previews: PreviewProvider {
    static var previews: some View {
        HomePageView()
    }
}
```

NextEvent

```
import SwiftUI
import SDWebImageSwiftUI

struct NextEvent: View {
   @ObservedObject private var postInfor = RecentProtestInfo23()

   @ObservedObject private var postInfor2 = RecentProtestInfo2()

   var body: some View {

      NavigationView{
         ScrollView{
            VStack{
               HStack{
                  Text("Upcoming protests")
                     .font(.title.bold())
                  Spacer()
               }
               LazyVStack{
                  VStack(alignment: .leading) {

                     WebImage(url: URL(string: postInfor.image1 ))
                        .resizable()
                        .aspectRatio(contentMode: .fill)
                        .frame(height: 220)
                        .frame(maxWidth: UIScreen.main.bounds.width -
80)
                        .clipped()
                        .clipShape(RoundedRectangle(cornerRadius: 20,
style: .continuous))
                     VStack(spacing: 6) {
                        HStack {
                           Text(postInfor.heading)
                              .multilineTextAlignment(.leading)
                              .fixedSize(horizontal: false, vertical: true)
                              .lineLimit(3)
```

```
                        .font(Font.title2.bold())
                        .foregroundColor(.primary)
                    Spacer()
                }
                HStack {
                    Text(postInfor.details)
                        .multilineTextAlignment(.leading)
                        .fixedSize(horizontal: false, vertical: true)
                        .lineLimit(10)
                        .font(.subheadline)
                        .foregroundColor(.secondary)
                    Spacer()
                }
            }
            .frame(height: 110)
        }
        .padding(15)
        .background(Color.white)
        .frame(maxWidth: UIScreen.main.bounds.width - 50,
alignment: .leading)
        .clipShape(RoundedRectangle(cornerRadius: 20, style:
.continuous))

        VStack(alignment: .leading) {
            WebImage(url: URL(string: postInfor2.image1 ))
                .resizable()
                .aspectRatio(contentMode: .fill)
                .frame(height: 220)
                .frame(maxWidth: UIScreen.main.bounds.width -
80)
                .clipped()
                .clipShape(RoundedRectangle(cornerRadius: 20,
style: .continuous))
            VStack(spacing: 6) {
                HStack {
                    Text(postInfor2.heading)
                        .multilineTextAlignment(.leading)
```

```
                            .fixedSize(horizontal: false, vertical: true)
                            .lineLimit(3)
                            .font(Font.title2.bold())
                            .foregroundColor(.primary)
                        Spacer()
                    }
                    HStack {
                        Text(postInfor2.details)
                            .multilineTextAlignment(.leading)
                            .fixedSize(horizontal: false, vertical: true)
                            .lineLimit(10)
                            .font(.subheadline)
                            .foregroundColor(.secondary)
                        Spacer()
                    }
                }
                .frame(height: 110)
            }
            .padding(15)
            .background(Color.white)
            .frame(maxWidth: UIScreen.main.bounds.width - 50,
alignment: .leading)
            .clipShape(RoundedRectangle(cornerRadius: 20, style:
.continuous))
                }
            }
            .padding(.horizontal, 15)
            .padding(.vertical, 30)
        }
        .navigationBarTitle("Protests")
        .onAppear(){

        }

    }
}
```

```
}

struct ViewA_Previews: PreviewProvider {
    static var previews: some View {
        NextEvent()
    }
}
```

ProfileView

```
import SwiftUI
import FirebaseFirestore
import MapKit
import SDWebImageSwiftUI


struct Live: Identifiable {
 var id: String
 var name: String
 var liveLatitude : Double
 var liveLongitude : Double
}


struct ProfileView: View {

    @State var showLogin = false
    @State var errorMessage = ""

   @ObservedObject private var vm = UserInfoModel()
    @ObservedObject private var postInfor = RecentProtestInfo()


    @ObservedObject private var viewModel = ProtestViewModel()

    var body: some View {



        NavigationView{
            ZStack{

                //image


                Circle()
```

```
    .scale(1.7)
    .foregroundColor(.white)

VStack{


    Text("Profile")
        .font(.largeTitle)
        .bold()
        .padding()
        .foregroundColor(.blue)
    Spacer()
        .frame(height: 50)
    WebImage(url: URL(string: vm.profileImage))
        .resizable()
        .scaledToFill()
        .frame(width: 200, height: 200)
        .clipped()
        .cornerRadius(44)
    Text(vm.email)
        .font(.largeTitle)
        .bold()
        .padding()
        .foregroundColor(.blue)

    Spacer()
        .frame(height: 50)
    Button {
        signOut()
    } label: {
        HStack {
            Spacer()
            Text("Sign Out")
                .foregroundColor(.white)
                .padding(.vertical, 10)
                .font(.system( size: 14, weight: .semibold))
                .frame(width: 300, height: 50)
```

```
                    Spacer()
                }.background(Color.blue)
            }
            .fullScreenCover(isPresented: $showLogin, content: {

                IntroView()
            })

            Text(self.errorMessage)
                .foregroundColor(.red)

            }

        }
      }
      .navigationBarHidden(false)
   }
   //Sign out function
   private func signOut() {

      try? FirebaseManager.shared.auth.signOut()
       if FirebaseManager.shared.auth.currentUser == nil {
          showLogin = true
       }

   }


}

struct ProfileView_Previews: PreviewProvider {
   static var previews: some View {
      ProfileView()
   }
}
```

AdminPage

```
import Foundation
import SwiftUI
import Firebase


struct AdminPage: View {

    @State var adminView = false
    @State var details = ""
    @State var name = ""
    @State var ID = ""
    @State var shouldShowImagePicker = false
    @State var image: UIImage?
    @State var errorMessage = ""


    @State var liveName = ""
    @State var liveDetails = ""
    @State var liveID = ""
    @State var startLocation = ""
    @State var startTime = ""
    @State var endTime = ""
    @State var endLocation = ""
    @State var liveLatitude : Double
    @State var liveLongitude : Double
    @State var note = "Placegolder"




    var body: some View {
        NavigationView{
            ScrollView{
                VStack(spacing: 16){
```

```
            Picker(selection: $adminView, label: Text("Picker
here"))  {
                Text("Next Event")
                    .tag(true)
                Text("Map")
                    .tag(false)
            }.pickerStyle(SegmentedPickerStyle())
                .padding()

            //Update next event
            if adminView{

                //Image Picker Button in order to choose a picture to
upload.
                Button {
                    shouldShowImagePicker.toggle()

                } label: {
                    VStack{
                        if let image = self.image {
                            Image(uiImage: image)
                                .resizable()
                                .scaledToFill()
                                .frame(width: 128
                                    , height: 128)
                                .cornerRadius(64)
                        } else {
                            Image(systemName: "person.fill")
                                .font(.system(size :64))
                                .padding()
                                .foregroundColor(Color(.label))
                        }
                    }
                    .overlay(RoundedRectangle(cornerRadius: 64)
                                .stroke(Color.black, lineWidth: 3))
```

```
    }
    Group{
        TextField("ID", text: $ID)
            .autocapitalization(.none)

        TextField("Title", text: $name)
            .autocapitalization(.none)
        Text("Description of the protest/ riot below")

        TextEditor(text: $details)
            .autocapitalization(.none)
            .frame( height: 250)
            .cornerRadius(10)
            .border(Color.black)


    }.padding(12)
        .background(.white)

}
//Update live map with information
if !adminView{
    Group{

        TextField("Tittle", text: $liveName)
            .autocapitalization(.none)
        TextField("ID", text: $liveID)
            .autocapitalization(.none)
        TextField("Start location", text: $startLocation)
            .autocapitalization(.none)

        TextField("Start time", text: $startTime)
            .autocapitalization(.none)
        TextField("End Location", text: $endLocation)
            .autocapitalization(.none)
        TextField("End Time", text: $endTime)
            .autocapitalization(.none)
```

```
                    Text("Description of the protest/ riot below")
                    TextEditor(text: $liveDetails)
                        .autocapitalization(.none)
                        .frame( height: 250)
                        .cornerRadius(10)
                        .border(Color.black)
                    }.padding(12)
                    .background(.white)

                Group{
                    Text("Latitude")
                    TextField("latitude", value: $liveLatitude, format:
.number)
                    Text("Longitude")
                    TextField("longitude", value: $liveLongitude,
format: .number )



                }.padding(12)
                    .background(.white)


            }

            Button {
                updateEvent()

            } label: {
                HStack {
                    Spacer()
                    Text(adminView ? "Update Next Event" : "Update
Map")
                        .foregroundColor(.white)
                        .padding(.vertical, 10)
                        .font(.system( size: 14, weight: .semibold))
                    Spacer()
```

```
                }.background(Color.blue)


            }


            Text(self.errorMessage)
                .foregroundColor(.red)
        }.padding()

    }
    .navigationTitle(adminView ? "Next Event" : "Map Update")
    .background(Color(.init(white: 0, alpha: 0.05))
                .ignoresSafeArea())
  }
  .navigationViewStyle(StackNavigationViewStyle())
  .fullScreenCover(isPresented: $shouldShowImagePicker,
onDismiss: nil) {
    ImagePicker(image: $image)
  }
 }


 //Toggle between update map and update next event
 private func updateEvent() {
   if adminView{
     self.EditNextEvent()
   }else {
     updateMap()

   }

 }
 //Store image that will be uploaded to firebase storage
 private func storeProtestInformation(imageProfileUrl: URL) {
```

```
        let adminData = ["ID": self.ID, "details": self.details, "image":
imageProfileUrl.absoluteString, "name" : self.name]
        FirebaseManager.shared.firestore.collection("Events")
            .document(self.ID).setData(adminData) { err in
                if let err = err {
                    print(err)
                    self.errorMessage = "\(err)"
                    return
                }

                print("Success")
            }
    }

    // Store all the information into firestore including image URL
    private func EditNextEvent() {
    //    let filename = UUID().uuidString
        //Current user UID

        let ref = FirebaseManager.shared.storage.reference(withPath:
self.name)
        guard let imageData =
self.image?.jpegData(compressionQuality: 0.5) else {return }
        ref.putData(imageData, metadata: nil) { metadata, err in
            if let err = err {
                self.errorMessage = "Failed to push image to Storage:
\(err)"
                return
            }
            ref.downloadURL { url, err in
                if let err = err {
                    self.errorMessage = "Failed to retrieve download URL:
\(err)"
                    return
                }
                self.errorMessage = "Successfully stored image with url
\(url?.absoluteString ?? "")"
```

```
            guard let url = url else {return }
            self.storeProtestInformation(imageProfileUrl: url)
        }
    }

}

// Update Map information in firestore
private func updateMap() {



    FirebaseManager.shared.firestore.collection("ProtestsLive")
        .document(self.liveID).setData(["id": self.liveID,"details" :
self.liveDetails, "name": self.liveName, "latitude": self.liveLatitude,
"longitude" : self.liveLongitude, "startLocation": self.startLocation,
"startTime": self.startTime, "endLocation": self.endLocation,
"endTime": self.endTime] ) { err in
            if let err = err {
              print(err)
               self.errorMessage = "\(err)"
               return
            }

            self.errorMessage = "Successfully updated map"

            self.liveDetails = ""
            self.liveName = ""
            self.liveLatitude = 0
            self.liveLongitude = 0
            self.liveID = ""
        }


}
```

```
}


struct AdminPage_Previews: PreviewProvider {
    static var previews: some View {
        AdminPage(liveLatitude: 0.0, liveLongitude: 0)
    }
}
```

ProtestInfo

```
import SwiftUI
import SDWebImageSwiftUI

struct ProtestInfo: View {
   @ObservedObject private var postInfor = RecentProtestInfo()


   @ObservedObject private var viewModel =
ProtestSubmittedViewModel()

   var body: some View {



      NavigationView{

         List(viewModel.info) { info in
            VStack(alignment: .leading) {
               Text(info.location)
                  .font(.headline)
               Text("\(info.dateTime)")
                  .font(.subheadline)
               Text("\(info.description)")
                 .font(.subheadline)







            }

         }
         .navigationTitle("Submitted protest")
         .onAppear(){
            self.viewModel.fetchInfo()
         }
```

```
        }
    }
}

struct ProtestInfo_Previews: PreviewProvider {
    static var previews: some View {
        ProtestInfo()
    }
}
```

SubmitInfo

```
import SwiftUI

struct SubmitInfo: View {
    @State var location = ""
    @State var dateTime = ""
    @State var description = ""
    @State var errorMessage = ""



    var body: some View {

        NavigationView{
            ScrollView{
                VStack{
                    Group{
                        TextField("Location including postcode and street
name", text: $location)
                            .autocapitalization(.none)

                        TextField("Date and time", text: $dateTime)
                            .autocapitalization(.none)
                        Text("Description")

                        TextEditor(text: $description)
                            .autocapitalization(.none)
                            .frame( height: 250)
                            .cornerRadius(10)
                            .border(Color.black)

                        Button(action: {
                            submit()

                        }, label: {
                            Text("Save".uppercased())
                                .font(.headline)
```

```
                    .foregroundColor(.white)
                    .padding()
                    .frame(maxWidth: .infinity)
                    .background(Color.blue)
                    .cornerRadius(10)
                })

                Text(self.errorMessage)
                   .foregroundColor(.red)



            }.padding(12)
                .background(.white)
        }
        .navigationTitle("Tell us about a protest")
        .font(.headline)
    }
}


}


private func submit() {

    let userData = ["location": self.location, "dateTime":
self.dateTime, "description": self.description]


FirebaseManager.shared.firestore.collection("ProtestSubmission")
        .document(self.location).setData(userData) { err in
            if let err = err {
              print(err)
               self.errorMessage = "\(err)"
               return
            }
```

```
            print("Success")
        }


    }
}

struct SubmitInfo_Previews: PreviewProvider {
    static var previews: some View {
        SubmitInfo()
    }
}
```

ProtyesApp

```
import SwiftUI

@main
struct appApp: App {
    var body: some Scene {
        WindowGroup {
          IntroView()
        }
    }
}
```

UserContentView

```
import SwiftUI

struct UserContentView: View {
    var body: some View {
        TabView{

            HomePageView()
                .tabItem(){
                    Image(systemName: "mappin.and.ellipse")

                    Text("HomePage")

                }.background(.white)
            NextEvent()
                .tabItem() {
                    Image(systemName: "note.text.badge.plus")

                    Text("Next Event")
                }

            SubmitInfo()
                .tabItem() {
                    Image(systemName: "plus")
                    Text("Protest")
                }

            ProfileView()
                .tabItem() {
                    Image(systemName: "person.crop.circle.fill")
                    Text("Profile")


                }
                .background(.white)
```

```
        }

    }
}

struct UserContentView_Previews: PreviewProvider {
    static var previews: some View {
        UserContentView()
    }
}
```

AdminContentView

```
import SwiftUI

struct AdminContentView: View {
    var body: some View {
        TabView{

            HomePageView()
                .tabItem(){
                    Image(systemName: "mappin.and.ellipse")

                    Text("HomePage")

                }.background(.white)
            NextEvent()
                .tabItem() {
                    Image(systemName: "note.text.badge.plus")

                    Text("Next Event")
                }

            SubmitInfo()
                .tabItem() {
                    Image(systemName: "plus")
                    Text("Protest")
                }
            AdminPage(liveLatitude: 0.0, liveLongitude: 0.0)
                .tabItem() {
                    Image(systemName: "a.circle.fill")
                    Text("Admin")
                }
            ProtestInfo()
                .tabItem() {
                    Image(systemName: "info.circle.fill")
                    Text("ProtestInfo")
                }
```

```
            ProfileView()
                .tabItem() {
                    Image(systemName: "person.crop.circle.fill")
                    Text("Profile")


                }
                .background(.white)


        }

    }
}

struct AdminContentView_Previews: PreviewProvider {
    static var previews: some View {
        AdminContentView()
    }
}
```

APPENDIX J: LINKS

This is the link to the GitHub where the work was pushed regularly:
https://github.com/SuadWarsame/Protyes

This is the link to the YouTube video where I put my video demonstration:
https://www.youtube.com/watch?v=ufkLOYgCq2M