**A logo for a university

Description automatically generated**

**Title: *Graduation Gathering***

**By**

***Adam Murray***

***Course: Computer Science BSc***

**Project code: *PJE40***

**Supervisor: *Dr Gail Ollis***

**April 2024**

**Word count: put final, estimated word count here**

**(should not include title page, acknowledgements, abstract, table of contents, table of tables, table of figures, reference list or appendices)**

Please tick

|  |  |
| --- | --- |
|  | I give permission for my project to be published in the University library and/or be made available to other students as examples of previous work. (optional). |
|  | I confirm that I have read and understood the University Rules in respect of plagiarism and student misconduct. |
|  | I declare that this work is entirely my own. Each quotation or contribution cited from other work is fully referenced. |
|  |  |

Date: 03/05/2024

Introduction

The problem this project aims to solve is that on graduation day, among the chaos and the crowds, it can be difficult to find other students and staff who you would like to talk to. This day for many could be the last opportunity to talk to those who you have connected with over the previous years. The solution Graduation Gathering hopes to provide is a real-time location sharing platform from which students and staff will be able to share their location and find those who they have connected with.

In order to connect as many students and staff as possible, Graduation Gathering must be available on both iOS and Android whilst providing safeguards so that the system cannot be abused. I intend to achieve this by restricting the use of the app to exclusively students and staff of the University of Portsmouth and only enabling location sharing on graduation day itself and in areas that are related to graduation. I also intend to ensure that users can clearly manage who has permission to see their location so that they can be sure that their location is only shared with those who they intend to share it with. Due to the nature of sharing your real-time location, the safety of the users is my first priority when making this application.

The idea of an app that has the ability to help people find one another is obviously not new, any messaging app can be used by individuals to arrange to meet up. However, this is extremely time consuming when done on a large scale and the chaos of graduation day would most likely lead to most of these messages not resulting in a meet up. Graduation Gathering would solve this problem by allowing users to select who they want to see the location of ahead of time, and then on the day they would just use the app to quickly find the other users.

For Graduation Gathering to be successful in its use, it will need to be both accurate in its location sharing and to be user friendly. Graduation Gathering is intended to only be used for a short time and so the UI must be intuitive or the user will simply not use the app.

This will require a system that has both the capability to display a map of Portsmouth and the locations of the users on said map, as well being able to handle location sharing for a large number of simultaneous users. This location sharing creates a problem as for the tracking to be accurate the locations of users will need to be updated very frequently and so the server will be called very often by the users clients. At the time, due to the nature of all the users having the potential to be sharing their locations with each other, the server must store the location data in a single place. The combination of these two factors means that all of the large number of frequent requests must, at some point, access the same point on the server making the server more likely to fail.

Methodology

The methodology I will use needs to be chosen, and altered, according to the specifics of this project. So to summarise, this project will be developed in its entirety by 1 developer, this project will require a front-end and a back-end, this separation allows different parts of the project to be completed separately, this project will also require a database. Also as the solo developer, I do not have a complete understanding of all of the technology that might be required. The last part of note is that the stakeholders of the app being its potential users, student and staff of the University of Portsmouth, as well as the graduation team for the University if they take an interest. Importantly, since this app is designed solely for graduation day, it is hard to get accurate feedback from the stakeholders as no one knows how it will actually be used on the day, until graduation day happens.

The work will be completed more efficiently if tasks related by how they are implemented are completed sequentially. For example, the database queries will be written more efficiently if written closer together in time due to the knowledge of the query language being fresh in my mind. Since there is only 1 developer, this means that we should try to use a methodology that allows us to focus on certain groups of tasks instead of implementing all of the project at once.

Since feedback is difficult to attain, but not impossible, for instance if a meeting with the graduation team is able to happen, we still need to allow flexibility within our methodology so that such feedback gain can still be acted upon.

So we need a methodology that allows us to update our requirements and design as we go along, whether from receiving feedback or from an improved understanding of the technology that is being used. The methodology must also allow us to work on related components of multiple features sequentially instead of demanding that a feature be completed before work can be started on another. For example, implementing the back-end of both the location sharing feature and the profile settings feature before going and implementing the front-end of either.

I considered Kanban, however with Kanban I would be required to write down all related tasks that would need implementing before I start the first task in a group. This means that as I complete the tasks, and so my knowledge of the areas grows, if I decide that I need to redesign a part of the system, I would then also need to redo the Kanban board as with even just a small redesign, the tasks would likely change. This will waste time, I believe it to be a better approach to keep future tasks in a group vague so that I can adapt easily to changes. I admit that in a group environment, this thinking ahead is required to keep all team members working in the same direction, and a Kanban board can then be an excellent way to do this. However, since I am working by myself, I would not gain this teamwork benefit and so Kanban is not suitable for this project.

I believe the most effective way to do this would be to use a modification of the Scrum software methodology. Scrum has teams complete work on related features in time-bounded sprints. Within a sprint I will choose part of the project to work on and I will aim to complete all tasks that fall under that part of the project. Examples of different sprint topics are as follows, the back-end of the project, the UI, the map interaction. I will modify Scrum by implementing sprints of varying time. Since not all of my topics will take a similar amount of time to complete, I will vary the length of my sprints accordingly. This approach allows me to iteratively build my project efficiently and alter the requirements and design at any point without causing major delays.

Requirements

For this project, as a student I have a good understanding of what I would want from an app marketed to graduating students. However, it is both good to get the opinion of other students as well as staff as this app is marketed towards to both students and staff. I also thought it was important to contact the graduation team as they are experts on graduation day. In order to get the opinions of students and staff I created a custom questionnaire that gave me the freedom to ask whatever questions I wanted to without being restrained by a tool such as Google Forms. As the for my contact with the graduation team, I arranged a 1 hour meeting with a member of the team during which I was able to get feedback on my ideas as well as talk about their ideas for my app. I have now split my requirements into functional and non-functional requirements and then ordered them using MoSCoW prioritization.

Graduation Gathering Questionnaire

This questionnaire targets students and staff, primarily asking them to rank a list of features from most important to least important. It does also ask about whether the person is a student or member of staff as well as which areas of Portsmouth they would like to become graduation zones, and finally asks for suggestions.

I received 17 responses to my questionnaire, 14 of which from students, 3 from staff.

For question 2, I asked the participants to rank a number features from which are most important to them to least important to them. These features are: Users of the app are verified University of Portsmouth staff or students; You can be selective about which graduation zones you appear in; You can be selective about which graduation zones you can see others in; you can filter which users you can see by their course; you can filter which users you can see by their name; you can filter which users you can see by whether you’ve already met; and finally, a messaging service that allows you to communicate with other users. The messaging service was ranked lowest on 88% of the responses. All the other features were pretty randomly mixed together throughout the responses.

For question 3, which posed 3 possible graduation zones (Guildhall Square, Ravelin Park, and the area between Guildhall Square and Ravelin Park) and asked for each of them whether users would like these areas to be zones or not, every single response said that all 3 of the zones should be zones.

For question 4, I asked as an optional question are there any other areas that the questionnaire taking would like to be designated as a graduation zone. I got 2 responses, one from a student asking the ability to set a larger area of Portsmouth as a zone for example 1 mile outside of the current zones. The other response from a member of staff asking to be able to draw their own zones.

For question 5, I asked as an optional question for any other ideas, for which I got no responses.

Graduation Team Meeting

I had a meeting with Lisa Scott, a member of the graduation team for the University of Portsmouth. During this meeting I showed them the app, as it was at the time, in order to get feedback, they also gave me ideas for future features. The ideas Lisa had include but are not limited to: showing key locations on the map such as photography at Ravelin Sports Centre and the ceremony at the Guildhall; a reminder sent to students before their ceremony reminding them to be at least 10 minutes early and that they cannot be late; a checklist of what students need to do on graduation day such as picking up their gown, attending their ceremony, going to the reception and collecting their certificate; adding a routing feature to help students both navigate to the above mentioned locations but also the other app users that they want to find. Lisa was also able to give me estimates on the number of students on site on a graduation day and so this can be interpreted as the maximum number of simultaneous users of the app, which would be 2000.

Functional

Must Have

|  |  |
| --- | --- |
| ID | Description |
| 1 | A map that is rendered on screen showing Portsmouth |
| 2 | The user’s current location displayed on the map when the user has location enabled |
| 3 | The ability for other users current locations to be displayed on the map to the user |
| 4 | The ability to manage who has permission to see your current location |
| 5 | The ability to manage who you have permission to see their current location |
| 6 | Users login with their university emails (@myport.ac.uk and @port.ac.uk) by having a code sent to their email that they will then enter into the app |
| 7 | All requests sent to the server encrypted using HTTPS |
| 8 | All requests sent to the server once a user had logged in being authenticated using a JWT(Json web token) generated by the server when the user is logging in |
| 9 | Designated graduation zones to appear on the map, only in which will users locations be shared |

Should Have

|  |  |
| --- | --- |
| ID | Description |
| 10 | Users locations only being shared on graduation days(8am – 2am the next day) |
| 11 | The ability for users to add their name to their account |
| 12 | The ability for users to add their faculty to their account |
| 13 | The ability for users to add their school to their account |
| 14 | The ability for users to add their course to their account |
| 15 | The account type (Student/Staff) saved to an account based on the email address that they used to login |
| 16 | The ability for users to select which graduation zones they must be in for their location to be shared |
| 17 | The ability to search for other users, when managing location permissions, by their email address |
| 18 | The ability to search for other users, when managing location permissions, by their name |
| 19 | The ability to search for other users, when managing location permissions, by their faculty |
| 20 | The ability to search for other users, when managing location permissions, by their school |
| 21 | The ability to search for other users, when managing location permissions, by their course |
| 22 | The ability to search for other users, when managing location permissions, by their account type |

Could Have

|  |  |
| --- | --- |
| ID | Description |
| 23 | The ability to tap users you can see on your map to display the information they have added to their profile |
| 24 | The ability to tap users you can see on your map and tick a box to say you have already seen this person and so to stop that user from continuing to display on your map |
| 25 | The ability to filter the users that appear on your map by faculty |
| 26 | The ability to filter the users that appear on your map by school |
| 27 | The ability to filter the users that appear on your map by course |
| 28 | The ability to filter the users that appear on your map by email |
| 29 | The ability to filter the users that appear on your map by name |
| 30 | The ability to filter the users that appear on your map by graduation zones |
| 31 | The ability to filter the users that appear on your map by account type |
| 32 | A checklist of what students need to do on graduation day such as picking up their gown, attending their ceremony, going to the reception and collecting their certificate |
| 33 | Markers on the map showing the key locations for graduation (Guildhall, Ravelin Sports Centre, Gun House Green) |
| 34 | The above mentioned markers clickable to display information about what is happening at these locations |

Will Not Have Right Now

|  |  |
| --- | --- |
| ID | Description |
| 35 | A routing feature to help users navigate to the above mentioned key locations |
| 36 | A routing feature to help users navigate to other users |
| 37 | The ability to draw your own graduation zones that you can appear in to others |
| 38 | The ability to message other users |
| 39 | The ability to send a push notification to another user to let them know that you’re trying to find them |

Non-Functional

Must Have

|  |  |
| --- | --- |
| ID | Description |
| 40 | The ability to run on the latest version of Android(Android 14) |
| 41 | The ability to handle 50 concurrent users without denying any of them service(Returning Service Unavailable when a user’s client calls an endpoint on the server) |
| 42 | A users location update on other users devices at most 30 seconds after the users client get an update of its own location (assuming all users have a good connection to the internet) |
| 43 | The JWTs that are generated by the server for users to login be valid for 24 hours |
| 44 | The login codes sent to users emails should be valid for 5 minutes |

Should Have

|  |  |
| --- | --- |
| ID | Description |
| 45 | The ability to run on the latest version of iOS(iOS 17) |
| 46 | The ability to handle 500 concurrent users without denying any of them service(Returning Service Unavailable when a user’s client calls an endpoint on the server) |

Could Have

|  |  |
| --- | --- |
| ID | Description |
| 47 | The ability to handle 2000 concurrent users without denying any of them service(Returning Service Unavailable when a user’s client calls an endpoint on the server) |

Design

For this project I have 4 pieces that need to come together, I need a map on which to display Portsmouth and all the users, the rest of the front-end of the app, the database, and finally the back-end server to link the database to the front-end.

For the map I will use the OpenLayers JavaScript library as I have a lot of experience with it using it for 2 previous projects. It is also free and allows me to both display a map, which will be an OpenStreetMap map, as well as overlay markers and shapes onto said map. Also since this is a JavaScript library it is compatible with any framework through the use of a WebView.

For the rest of the front-end of the app, I will use Flutter as it allows me to develop both for iOS and Android whilst writing just one set of code. Flutter also has a very large set of libraries developed for it that will allow me to easily and efficiently extend the functionality of the app. Finally, I have experience using Flutter to develop an App as I have previously used it to create my app University Bus Portsmouth.

The server will be hosted on AWS as I have experience using this tool and it allows me to efficiently create a database using Amazon RDS and create AWS lambda methods to add additional features. AWS also has an email service that I can utilise to send the login codes.

The database will be a MySQL database as this is the type that I am most familiar with and it has the capabilities that I require.

I have split the rest of the design of this project into 3 distinct sections: the UI and visual design; the systems architectural design; and finally the database design. Whilst these are presented as separate, the design was not done separately, instead an iterative approach was taken to the design when adding each feature. A part of the UI would be designed to display the feature to the user, the architectural design was then updated to support this feature, and finally the database was updated as necessary to complete the design of the feature from front end to back.

UI and Visual Design

Since this is an app for the University of Portsmouth, the aim is to use the colours of the university, which are primarily a dark purple and blue.

To create initial designs for the UI I will use Flutter Viz as it is a graphical tool meant for building UI to use for Flutter projects. Due to this the designs used in the project itself should be similar, but will have to be recreated out of Flutter code.

Login Screen

The login screen needs to allow users to enter their email and then enter the code sent to their email. So first I will display a text box and text field allowing users to enter their email. Then only once that is done I will display a box for the code to be entered. The background will be coloured in purple.

A computer screen shot of a email box

Description automatically generated A screenshot of a computer

Description automatically generated

Main Screen

The main screen will be divided into three sections, the app bar at the top coloured in purple, the navigation bar at the bottom, and the content of the current screen in the middle.

A black rectangle with white dots

Description automatically generated

In this case since the current screen is the home screen which contains a Map at the centre which cannot be replicated in Flutter Viz, this Map has been left blank. The map displayed will be an Open Street Map with markers and shapes overlaid.

Connections

The connections screen should allow users to manage who has permission to see them, and who they have permission to see. To allow this, the connections screen will first display a list of all the users that this user either has permission to see, or that has permission to see this user. I will also supply buttons next to each user in the list so the given user can manage whether they should have permission or not.

I will also have a button at the top that takes the user to a screen that will let them request permission to locate other users. This screen will then allow users to search for other users and provide a button to request permission from them.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Profile

The profile screen should allow users to enter their name, faculty, school, and if applicable, course. It should also allow users to select which graduation zones they want to be seen in.

A screenshot of a cell phone

Description automatically generated

Architectural Design

The architecture can be split into 2 parts, the front-end and back-end. The back-end will be created out of a set of AWS lambda methods that will retrieve data from the database and set data into the database. All of these lambda methods, except the login methods, will be authenticated with a JWT(json web token) that will be generated by the server when the user logs in. The use of JWTs will allow all requests sent to the server to be associated with the user that sent the request.

The front-end design will be based around the different screens as there will not be a significant part of the front end that is not directly working with the UI. The UI will be built by embedding screens(widgets) within other screens(widgets), this is standard for Flutter. However, for this project, all functionality will be to its relevant screen, and if its needed by multiple screens, then it is attached to the closest parent screen to both of the child screens.

A diagram of a company

Description automatically generated

Database

The database is a MySQL database hosted on AWS using the RDS. AWS was chosen for this project as they have a very reliable database service that I have used before whilst also allowing for back-end processing to be written and deployed efficiently using AWS lambda.

I used https://app.diagrams.net/ to create an entity relationship diagram for my database. There are five parts of this schema, 4 of which are linked by the user table, the location permission part, the course part, the graduation zones part, and finally the user table itself. The one other part is the Graduation Dates table which is on its own an just holds the days on which graduation ceremonies are held.

A screenshot of a computer

Description automatically generated

User Table

This table stores everything directly related to a user, which includes data such as their name and email address, but also their course, their last known latitude and longitude along with the time when that location was set, as well as their login code. When each user first enters their email address a unique user id is generated for them and that is set as the primary key in the database so that in the future if support for changing your email address is desired then it will be more feasible to add.

The has\_logged\_in row and the email\_verified are separate as has\_logged\_in is used to direct the app to take the user to their profile if they have never edited it before and so remains false until the profile has been edited, whilst email\_verified is false until the user correctly enters the code which will log them in for the first time.

Location Permissions

The location\_permission table is used to give permission to see other users. When a user, let’s say User A, first asks for permission from another user, User B, a new entry gets added to the location\_permission table where from\_user is the user id of User A, to\_user is the user id of user B, and permission\_granted is set to Requested. Then when user B accepts the request, the permission\_granted row is changed to Granted. This now allows for User A to see the location of User B, but not the other way around. If User B wants to see User A then they must request permission separately, at which point a new entry will be created in the location\_permission table where from\_user is the user id of User B, to\_user is the user id of user A, and permission\_granted is set to Requested.

This approach of creating separate entries in the location\_permission table for when User A requests permission from User B and from the other way around allows for location permissions to be managed separately, as in it allows User A to see User B but not the other way around, whilst still keeping an efficient and easy to understand database design.

Course

The faculty, school and course tables do not store any information about the users themselves. What these tables do store is the academic structure of the university, all the courses on offer, linked to which school they are in, and linking each school to which faculty they are in. This then allows the user table to efficiently store the faculty, school and course of the user by simply linking to these tables.

The advantage of this is that when a course, school or even faculty is removed from the University of Portsmouth, its entry into its table can simply be deleted and its presence in the entries of the users who have added it to their profile and easily be deleted as well by cascade.

Graduation Zones

The actual geojsons for the graduation zones are stored in the graduation\_zones\_text table whilst the label for the zone is stored in the graduation\_zones table. These are separate due the fact that the geojsons are too large to be indexed in a MySQL database, and so the solution to this problem is to store the zone id and label in a separate table that can then be indexed for fast retrieval. These zones are then linked to users using the user\_zones table.

When a user saves to their profile that they want to appear in a given zone, then an entry in the user\_zones table is created with both the users id and the zones id. When a user then saves that they no longer want to appear in a zone, that entry is then deleted. This allows for quick retrieval of which zones a user wants to appear in, not getting slowed down by the lack of indexing in the graduation\_zones\_text table.

Verification and Validation

Verification is checking that the software does what we expect it to do without having any bugs. Validation is checking that the software does what the requirements ask it to do.

For this project, at each stage of the implementation, I will first consider validation as it is pointless to test and debug only to realise afterwards that you built the wrong thing. I will think through the feature that I want to add, both before and after adding it, to ensure that it is what is required. Remember, the key goal for this project is to make it easier for students and staff to meetup on graduation day, so I must be convinced that a feature will help to do this in order to add this feature.

Once a feature has passed initial validation, I will then begin the implementation. In order to verify a feature it will undergo 2 sets of testing, manual debugging and automated unit testing. The server will also undergo load testing in order to verify it can handle the expected number of concurrent users.

Manual Debugging

As I implemented every feature, I tested them extensively with a variety of inputs to verify that the software is acting as expected. This level of testing focused on the larger scale of the feature and so did not send specific inputs to the individual methods. However, this level of testing does include interacting with the UI and making sure that it reacts to user input appropriately, but also testing the server endpoints to make sure they return current data from the database. This level of testing uses the live server and not a mock of the methods and so could be unpredictable in exactly the results presented, however is reflective of how the app would be used and so gets a good perspective on the app. To be clear, this perspective is not as good as the feedback that would be given by actual users.

Automated Unit Testing

For each public method implemented in the client side of the app, barring a few exceptions, unit tests have been written to automatically give the method a variety of inputs to check it returns the expected result. For this level of testing, mocks have been used to return data from the database and from local storage, this is to ensure consistency in the inputs to the test. Due to this reason any method that’s sole purpose is to retrieve data from the server unedited, has not had tests written for it as these tests would not accomplish anything. Unit testing is rigid and predictable and doesn’t reflect how actual users would interact with the program, however it does help you check you haven’t inadvertently broken an existing part of the program. The act of writing unit tests also makes you think about what the software is doing and how it accomplishes its task, this both helps with validation and verification. There have been many times when I have fixed a bug I didn’t know existed through writing unit tests, not through the output of the test being wrong, but through getting another perspective of the code written.

Load Testing

In order to test if the server can support the desired number of concurrent users, I have designed a load testing experiment that can be run efficiently and repeated as the server is updated. I have written a poster on this experiment and the results that I got when I conducted the experiment. To summarise the findings of this experiment, here is a quote from the conclusion: “This performance evaluation has shown that the Graduation Gathering server functions properly up to 300 concurrent users and seamlessly up to 70 concurrent users when subjected to realistic test conditions. However, this is far from the 2000 concurrent users that the Graduation Gathering server is expected to handle.”

Evaluation

The aim of this project was to create a location-sharing application to help students and staff of the University of Portsmouth connect on graduation day. In order to do this I have developed an app for both iOS and Android using Flutter with a server hosted on AWS. I have thoroughly tested each feature implemented to ensure it is bug free.

In this section I will evaluate my artefact against the requirements that I set out to implement. I will go through each requirement listed in the requirements section of this document and analyse whether it has been accomplished or not. I will then evaluate to what extent the aim of this project has been achieved based on the completed requirements.

In the tables of requirements below I have coloured in the rows of the completed requirements green, of the non-completed requirements red, any requirements where it is unknown whether it has been met shall be coloured orange, and finally any requirements where it has been partially met will be coloured blue and discussed further below.

Functional

Must Have

|  |  |
| --- | --- |
| ID | Description |
| 1 | A map that is rendered on screen showing Portsmouth |
| 2 | The user’s current location displayed on the map when the user has location enabled |
| 3 | The ability for other users current locations to be displayed on the map to the user |
| 4 | The ability to manage who has permission to see your current location |
| 5 | The ability to manage who you have permission to see their current location |
| 6 | Users login with their university emails (@myport.ac.uk and @port.ac.uk) by having a code sent to their email that they will then enter into the app |
| 7 | All requests sent to the server encrypted using HTTPS |
| 8 | All requests sent to the server once a user had logged in being authenticated using a JWT(Json web token) generated by the server when the user is logging in |
| 9 | Designated graduation zones to appear on the map, only in which will users locations be shared |

Should Have

|  |  |
| --- | --- |
| ID | Description |
| 10 | Users locations only being shared on graduation days(8am – 2am the next day) |
| 11 | The ability for users to add their name to their account |
| 12 | The ability for users to add their faculty to their account |
| 13 | The ability for users to add their school to their account |
| 14 | The ability for users to add their course to their account |
| 15 | The account type (Student/Staff) saved to an account based on the email address that they used to login |
| 16 | The ability for users to select which graduation zones they must be in for their location to be shared |
| 17 | The ability to search for other users, when managing location permissions, by their email address |
| 18 | The ability to search for other users, when managing location permissions, by their name |
| 19 | The ability to search for other users, when managing location permissions, by their faculty |
| 20 | The ability to search for other users, when managing location permissions, by their school |
| 21 | The ability to search for other users, when managing location permissions, by their course |
| 22 | The ability to search for other users, when managing location permissions, by their account type |

Could Have

|  |  |
| --- | --- |
| ID | Description |
| 23 | The ability to tap users you can see on your map to display the information they have added to their profile |
| 24 | The ability to tap users you can see on your map and tick a box to say you have already seen this person and so to stop that user from continuing to display on your map |
| 25 | The ability to filter the users that appear on your map by faculty |
| 26 | The ability to filter the users that appear on your map by school |
| 27 | The ability to filter the users that appear on your map by course |
| 28 | The ability to filter the users that appear on your map by email |
| 29 | The ability to filter the users that appear on your map by name |
| 30 | The ability to filter the users that appear on your map by graduation zones |
| 31 | The ability to filter the users that appear on your map by account type |
| 32 | A checklist of what students need to do on graduation day such as picking up their gown, attending their ceremony, going to the reception and collecting their certificate |
| 33 | Markers on the map showing the key locations for graduation (Guildhall, Ravelin Sports Centre, Gun House Green) |
| 34 | The above mentioned markers clickable to display information about what is happening at these locations |

Non-Functional

Must Have

|  |  |
| --- | --- |
| ID | Description |
| 40 | The ability to run on the latest version of Android(Android 14) |
| 41 | The ability to handle 50 concurrent users without denying any of them service(Returning Service Unavailable when a user’s client calls an endpoint on the server) |
| 42 | A users location update on other users devices at most 30 seconds after the users client get an update of its own location (assuming all users have a good connection to the internet) |
| 43 | The JWTs that are generated by the server for users to login be valid for 24 hours |
| 44 | The login codes sent to users emails should be valid for 5 minutes |

Should Have

|  |  |
| --- | --- |
| ID | Description |
| 45 | The ability to run on the latest version of iOS(iOS 17) |
| 46 | The ability to handle 500 concurrent users without denying any of them service(Returning Service Unavailable when a user’s client calls an endpoint on the server) |

Could Have

|  |  |
| --- | --- |
| ID | Description |
| 47 | The ability to handle 2000 concurrent users without denying any of them service(Returning Service Unavailable when a user’s client calls an endpoint on the server) |

As you can see from the above tables, all but one of the requirements I marked as a must have requirement, have been successfully implemented. Further, all of the functional should have requirements have also been implemented. However, none of the functional could have requirements have been met. For the non-functional requirements in the should have and could have categories, the server has failed to meet expectations and cannot handle the maximum expected number of concurrent users.

However, note that for requirement 45, this requirement has been marked as unknown as although the app has been built using Flutter and only contains packages that can be used on iOS, in order to actually compile and so test the software on iOS, you require an apple computer which I have been unable to acquire. Due to this the software most likely works on iOS but is completely untested and so this requirement cannot be verified.

For requirement 6, this has been partially met however does not fully work as intended. The system correctly only allows for accounts to be made using university emails. However, the server uses AWS Simple Email Service to send emails, and whilst the code I have written works currently to send the email, my AWS account is restricted to only being able to send emails to reverified email accounts. In other words, I can send emails to myself and only myself. I have looked into alternative email services however I do not believe any of them would allow me to use them.

Conclusion

Graduation Gathering does successfully allow for seamless location sharing between up to 70 concurrent users on Android devices. It also does this securely encrypting and authenticating all requests made to the server. Users are also able to efficiently manage who they want to have permission to see them and who they have permission to see. Finally users are also able to customise their profile if they wish. This is a major success and should not be understated as this means that Graduation Gathering has achieved the goal it set out to achieve.

However, it lacks some quality of life features that would make it easier to use, such as being able to filter who displays on your map at a given time and being able to tap users on the map to see who they are.

Graduation Gathering’s UI is also not the prettiest, this is not something that can be judged by requirements. Whilst it does stick to a nice colour scheme based off of the University of Portsmouth’s colour, the look of the buttons and text is not the nicest in places.

Finally, the server does not meet expectations. It does allow for the must have required 50 concurrent users which does make the app usable under its current state. However, if Graduation Gathering were ever to be rolled out to actually be used and endorsed by the graduation team at the University of Portsmouth, it would need to be far more capable than it currently is.

Overall, this project is a success, however too many requirements were taken on for the time frame given. The main problem was unforeseen events affecting production far more than I anticipated.

Conclusions and Future Work