Deliverable 2,
Software Architecture

University of New South Wales
Requirements & Design Workshop (SENG2021)
by Team BugFreeSince93

2020, Term 1

Andrew Han	**z5204829**
Harikrishna Manogaran	**z5124767**
Niraj Sapkota	**z5259070**
Omar Lotfi	**z5265983**
Timothy Thacker	**z5115699**

1.0 Problem Statements

1.1 The Problem Contextualized

The problem we aim to address is the difficulty of socialising and making friends at University. The problem mostly is observed in students that are transitioning from high school into university. However, this problem is not limited to high school students as international students also face difficulty making friends. They have often left their families and old lives behind to attempt to build a better life but face difficulty adjusting and coping with the new environment that confronts them. International students make up ~38% of the student demographic in UNSW (UNSW Annual Report 2018)[1] so it is a considerable problem. Furthermore, for local students, it is comparatively easier to make friends in high school as students have shared classes and break schedules. This in conjunction with the fact that they are not taught advanced social skills means they may face difficulty in university. This is because in university, making friends requires significantly more effort as everyone works to a different schedule. Thus, as we can extrapolate, this problem is affecting the entire student body of university - both local and international students face difficulty making friends either due to a lack of social ability or being put in an environment drastically different than what they were accustomed to.

1.2 The Purpose and Uniqueness of our System

We want to create an application to increase the accessibility and ease of making friends at University. We wish to do so using society events as a medium to match people with similar interests so they are able to find someone who they can relate to. This will ensure that they do not feel overwhelmed, which might be the case if they go alone. Having a friend will also provide a greater sense of social confidence which will allow them to socialise and fit in easier. Thus, our proposed system is an integration of other existing systems such as Facebook, Tinder, Linkup, Meetup and Timeweave. We plan to take the strengths of each platform and utilize them to solve our specific problem by providing a platform where users are able to find new friends easily.

[1] UNSW Annual Report 2018
https://www.unsw.edu.au/sites/default/files/uploads/UNSW%20Annual%20Report%202018%20170120.pdf

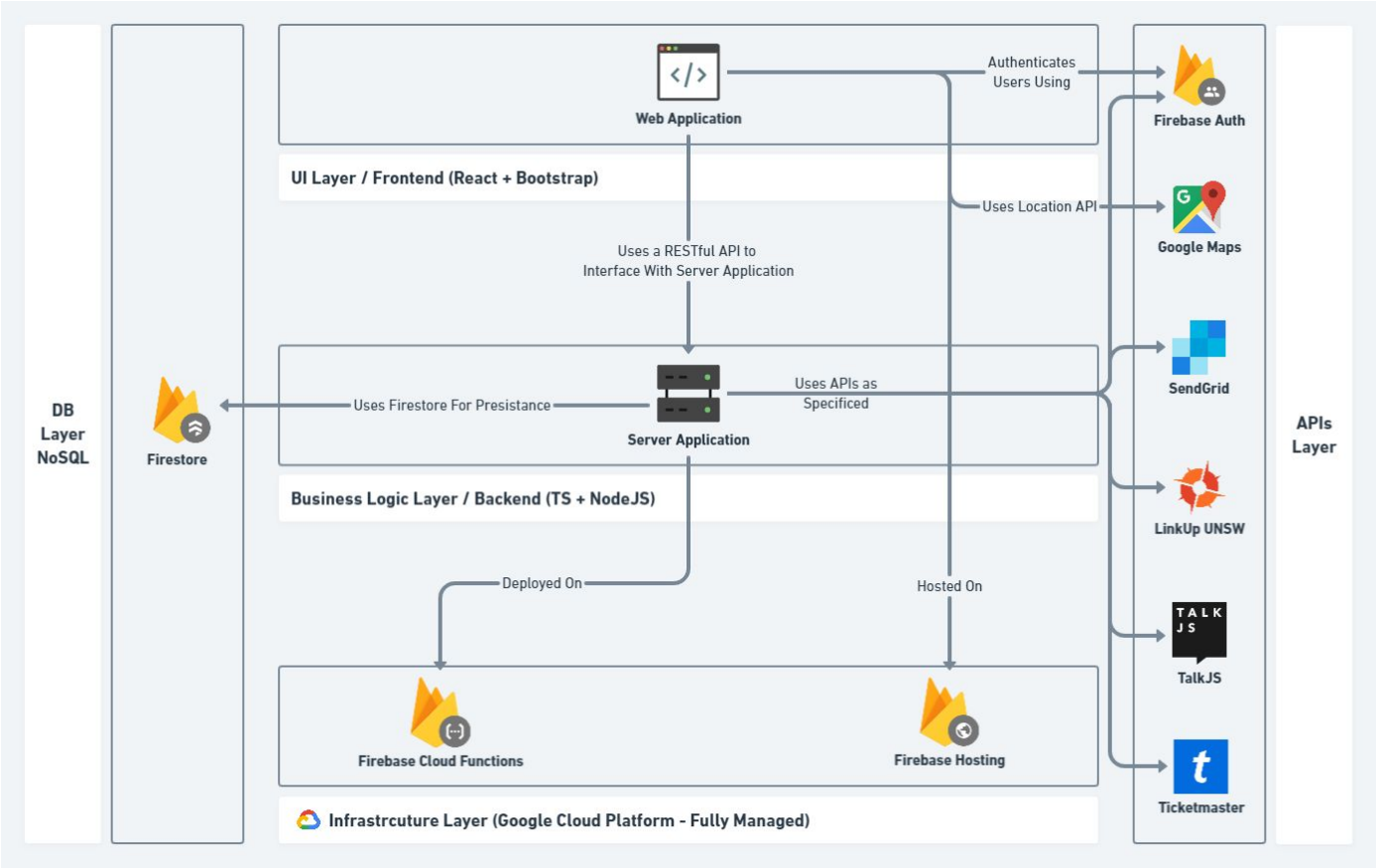
1.3 Summary Statements

To reiterate the problems that must be addressed and provide a clear guide towards the features we should implement, summary statements are listed below:

1. Both domestic and international students struggle to make friends in university.
2. Students find it difficult to approach and talk to new people without a stimulus.
3. It is hard to find people to go to society events with, and it can be intimidating going by oneself.
4. There is no highly integrated system that allows you to make friends at university.
5. These systems should be smart and find people that hold similar interests and are in the same area of study.
6. If this problem of social ineptitude is not addressed it extends beyond university and into adult life.
7. Friendships are vital to health, wellbeing and especially during university which can be one of the most stressful times of a person's life.
8. Without help, students continue to struggle to refine their interpersonal and social skills. Skills, which are critical for succeeding in industry.

2.0 Software Architecture

2.1 Architecture Diagram



2.2 Analysis of Software Architecture

Web Architecture

Our web application follows a versatile and modern web design architecture that allows for maximum flexibility and minimum coupling. Firstly, We will be building our server application to expose a RESTful JSON API. Secondly, our client application (web app) will be a Single-Page Application (SPA) that utilises the JSON API. This solution was chosen over others like multi-page server-side rendered applications because of its many advantages and benefits. This architecture allows the separation of the backend and frontend so they can be developed concurrently by different team members. It encourages the decoupling of business logic and the user interface in our application. It provides a fast and smooth experience for the user since the client application has full control over the data it requests over the network.

Infrastructure Layer

Our RESTful JSON API requires a server to run and respond to user requests. For example, a user may input an email and password and the server will respond to that event by validating the inputs. Typically, requests are sent to a server that you manage. However, this usually requires an upfront cost to buy the hardware and has extra costs (electricity, updates) to maintain the server for 24/7 use. It also does not scale well, as extra servers may be needed if the server capacity is full. The alternative is using a serverless architecture, which provides computing resources as they are needed, rather than having a constantly running server. All the server space and infrastructure is maintained and handled by a third party vendor. Also, it is charged based on the services that are used and the number of computations. For a small scale project such as ours, it would be cheaper to use serverless applications as there would not be too many executions and it would be faster to develop, as the infrastructure is handled by the vendor.

For our project, we will be using Google's serverless 'Cloud Functions for Firebase'. "Cloud Functions for Firebase let you automatically run backend code in response to events triggered by Firebase features and HTTPS requests." (Google, 2020)[2]. Although there are other alternative products that work in the same way, such as Amazon's 'AWS Lambda', 'Cloud Functions for Firebase' easily integrates other Firebase features that we will use, such as 'Firebase Authentication' and 'Firebase Firestore'. 'AWS Lambda' can integrate these features, but it will take more effort.

Our web application consists of static HTML, CSS, and JavaScript files that are served over the network to the user. Every time the user navigates to the web app URL, the browser downloads and executes these files. To ensure maximum speed and minimise the loading time for our users, we chose Firebase Hosting to host these static files. Firebase Hosting provides a global Content Delivery Network (CDN) built on top of the Google backbone network which has edge servers all around the world. The user gets these static files from the nearest one to their location to minimise latency. Firebase Hosting also provides automatic SSL certificates for security, easy deployment and rollback. This provider was chosen over hosting these files on our own servers, and other static hosts like Netlify and Github Pages due to it's deep integration with other Firebase services and superior features/performance.

[2] "Cloud Functions for Firebase - Google." 3 Dec. 2019, https://firebase.google.com/docs/functions. Accessed 24 Mar. 2020.

Database Layer

A database is a collection of data or information that is organised in a way to store, manage and retrieve information. An example of stored information could be a user's personal details. Databases are usually stored on database servers, which follow similar pros and cons in the interface layer above. Third party providers can provide storage for databases or developers can build their own hardware for storing data. For our project, we are using 'Firebase Firestore', a scalable database that can sync data across different apps. It can be easily integrated into the previously mentioned 'Cloud Functions for Firebase'. It is useful to us as our app 'Holler' can be used for both desktop and mobile users. If a user uses both devices, the ability to sync data in real-time can be helpful to us and the users. Firestore also automatically scales based on our requirements; pricing is based on how often the data is accessed and the amount of data stored.

There are alternative products which have similar functionalities, such as Amazon's DynamoDB. However, DynamoDB does not sync devices in real-time like Firestore. Also, DynamoDB's payment system is based on the amount you request; it does not automatically scale like Firestore. As we are not sure on how much traffic our application will generate, Firestore is suitable for our small application. However, in the future, if more users use the application, it will automatically scale the database for more users automatically.

The database is accessed by a NodeJS client library provided by Firebase only on our server application and is not accessed from the web application directly.. This is to ensure only authenticated and authorised users get access to the right data. While security mechanisms exist in Firebase like 'Firestore Security Rules' that could allow for direct access from the client, we chose to limit the database access to the server application to ensure the decoupling of our server/client applications.

API Layer

Application Programming Interfaces (APIs) are programs that offer services to other programs. For example, a developer can use authentication APIs to verify a user's inputs on their website. Our team will be utilising various APIs and other external data sources for our application. For login and signup authentication, our team will use Google's Firebase Authentication API. As another Firebase service, it will also be integrated seamlessly with other Firebase products we have mentioned. It provides everything related to signing in and logging in. It allows users to sign up and log in with email addresses, passwords and even phone numbers, or through Google Log-in or Facebook Login. It also provides two factor authentication to verify that an email belongs to the user that is signing up. Firebase delegates authentication/user management to ensure security and proper implementation. There are many other authentication APIs, such as Amazon Cognito or Auth0. However, we have chosen Firebase Authentication, as it is another Firebase product, which can be easily integrated with other Firebase products mentioned. Amazon Cognito and Auth0 may require more setup and may be more complex to integrate. Firebase Authentication also provides syncing across multiple clients in real time. As events and users are added to the application, our application would update the changes instantly, rather than having to update manually.

As our app displays information related to society events within the university and events outside the university, we will be using LinkUp UNSW API and Ticketmaster API for this purpose. These APIs all provide ways of searching for events within UNSW, as well as events outside of UNSW. There are many other event APIs, but these 2 were chosen as they are the biggest APIs that allow searching for events without any restrictions. Other options included, the Eventbrite API which has a large selection of events, but does not support searching for events. And the Songkick API which requires an application for an API key before a developer can search for events.

Our app also allows users to chat with other users, and we will be using the TalkJS API to build a fully featured chat. TalkJS features integrated messaging, which allows developers to add user-to-user chat to a web application or platform. It has been chosen over other popular chat APIs such as SendBird or OpenTok, as our application will not and should not feature voice calls or video chats. SendBird and OpenTok does have these features, but our application is focused on making friends based on interests rather than appearances, thus, a voice or video chat video is unnecessary. TalkJS only features a text chat which is the only form of communication needed on our app, and the lack of extra features makes it easier to use.

If a user decides to allow email notifications, it will be handled through the SendGrid API. SendGrid is a cloud based service for sending emails. It allows email newsletters, shipping notifications and more importantly, sign-up confirmations. Although there are simpler alternatives such as MailChimp, SendGrid features the ability to track and evaluate marketing statistics such as how many emails were opened or sent to spam. This is useful as it can score how active a user is, which would be displayed in a user's profile. It also has one of the most generous free trials, allowing 40,000 emails for the first 30 days.

We will use Google Maps API to track the locations of each event. It provides Street View imagery and allows users to find routes to a specific location. Mapbox was an alternative that we considered, but it has features that are ultimately unnecessary, such as creating custom online maps. Due to the widespread usage of Google Maps and Google products, people would be most familiar with the Google Maps layout.

Other data sources that we will use is the UNSW Timetable, which features all the subjects in UNSW. This allows users to put in their profile the subjects they are taking for the term. There are unfortunately no better alternatives than from an official UNSW website.

All the above mentioned APIs are accessed on our server application to ensure users' security and proper separation of concerns. We do not access any API from the frontend except for 2 APIs as illustrated in the architecture diagram. The first one is the Firebase Authentication API. Our frontend calls this API to log in users directly without interfacing with our servers since this API abstracts the whole JWT, stateless authentication process so it's required to be done on Firebase's servers. The second API is Google Maps API which is used from the frontend to display the location and directions to the events' venues for the user. It's not used from the server application since it's purely user interface data delivered directly from the Maps API using their JS SDK.

Business Logic Layer / Backend

The technology stack used for the backend will be:

- Language: TypeScript
- Runtime: NodeJS
- Web Framework: ExpressJS
- Persistence: Firestore NoSQL Database..

We have chosen Node.JS and React JS as the main backend and frontend languages as it allows for "Full Stack JavaScript". Having JavaScript on both the backend and frontend makes development faster and also allows us to only focus on one overall language. Our team is also familiar with Javascript overall, as all of us have experience using Javascript before. Also, using Javascript allows us to make use of the large ecosystem out there through NPM; thousands of free packages can be downloaded and used.

We have considered using alternative languages. For example, for the backend, we have discussed using Java or Python, as they are also commonly used for backend development. However, Python is not an application built for mobile or desktop application development. There are few libraries compared to Javascript which has numerous libraries tailored towards developing applications. It is possible to build applications using Java, but it is a more general usage language. Compared to Javascript, which again, is tailored towards application development, Javascript is more suitable towards our project. There were also considerations for the database language for Firestore. We went with the default NoSQL, as it is highly scalable, and there are well designed development tools made for faster development. We considered Firebase FireSQL for querying our database, but it does not allow the deletion or updating of data.

User Interface Layer / Frontend

The technology stack used for the frontend will be:

- JavaScript Framework: ReactJS
- CSS Framework: Bootstrap (for better responsive design)

For the frontend, we have considered using Angular or just plain HTML. However, Angular is a Javascript framework compared to React which is just a Javascript library. The learning curve is steeper and could hinder development as concepts such as the Model View Controller (MVC) architectural pattern need to be learned. MVC is only optional in React. Also, using plain HTML is difficult as it requires us to manually read or update the DOM, which is a representation of the page that a user sees. Bootstrap was used alongside React as it is the most popular CSS framework; there is extensive documentation and help online, compared to smaller frameworks such as Foundation or Pure.

A user of our application would access it through their web browser. The 5 most common browsers are Chrome, Firefox, Internet Explorer, Edge and Safari, which takes up ~93% of the browser market share. (NetMarketShare, 2018)[3] Testing will be done to ensure that these browsers are compatible with the 5 main browsers. As the rest of the market is made up of many other browsers, it would be a waste of time and resources to test for all of them.

[3] "Browser market share - NetMarketShare." https://netmarketshare.com/browser-market-share.aspx. Accessed 24 Mar. 2020.

3.0 User stories

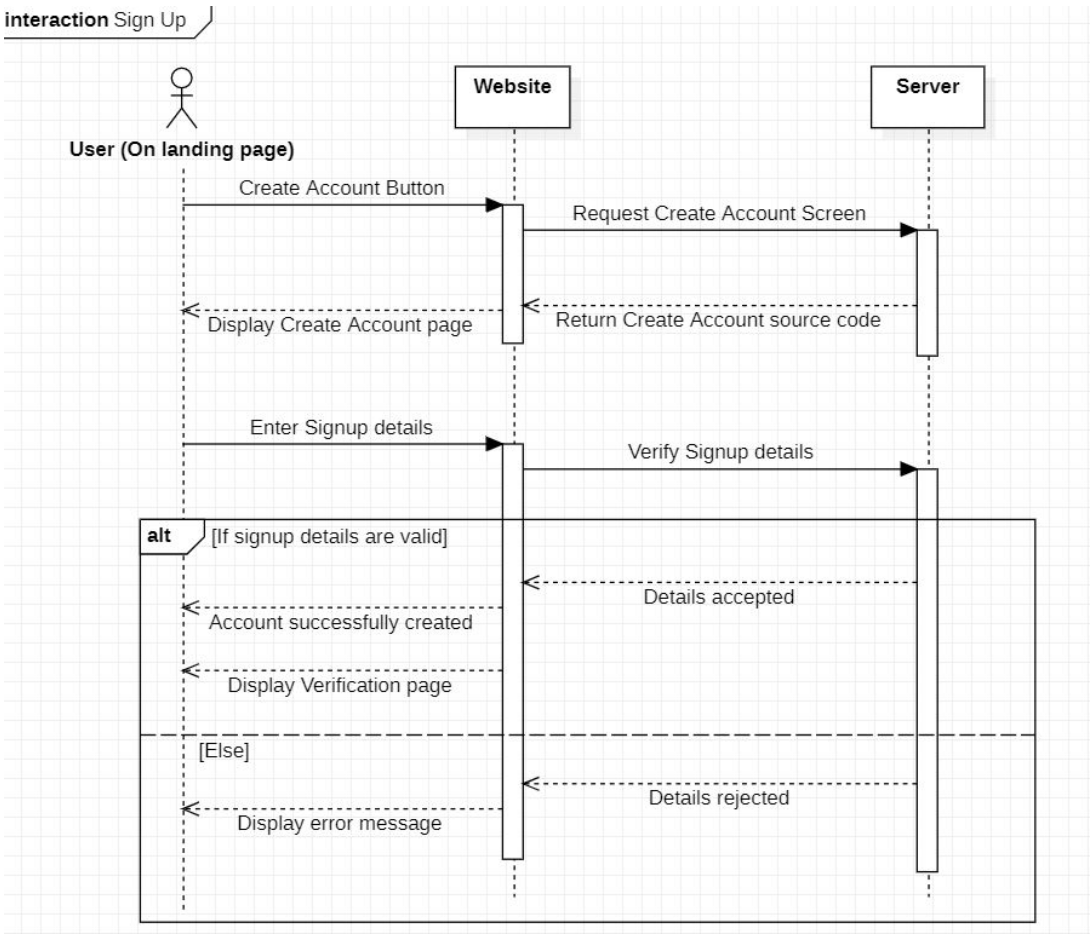
3.1 Sign Up / Sign In / Onboarding

Epic 1: As a student, I want to sign up and log into Holler so that I can start using the app

1. As a student, I want to sign up for Holler so that I can start making friends.

- Given a user is in the landing page, a ‘Sign Up’ button is available
- Given a user is on the sign up page, then a user should be able to input an email, password and confirmation password
- When a user puts in a password and confirmation password, then they must match
- Email must be valid and must not have been used to create a prior account
- When an invalid input is put in, then an error message must be printed
- When all the inputs are filled and they are valid, then a confirmation email is sent to the user

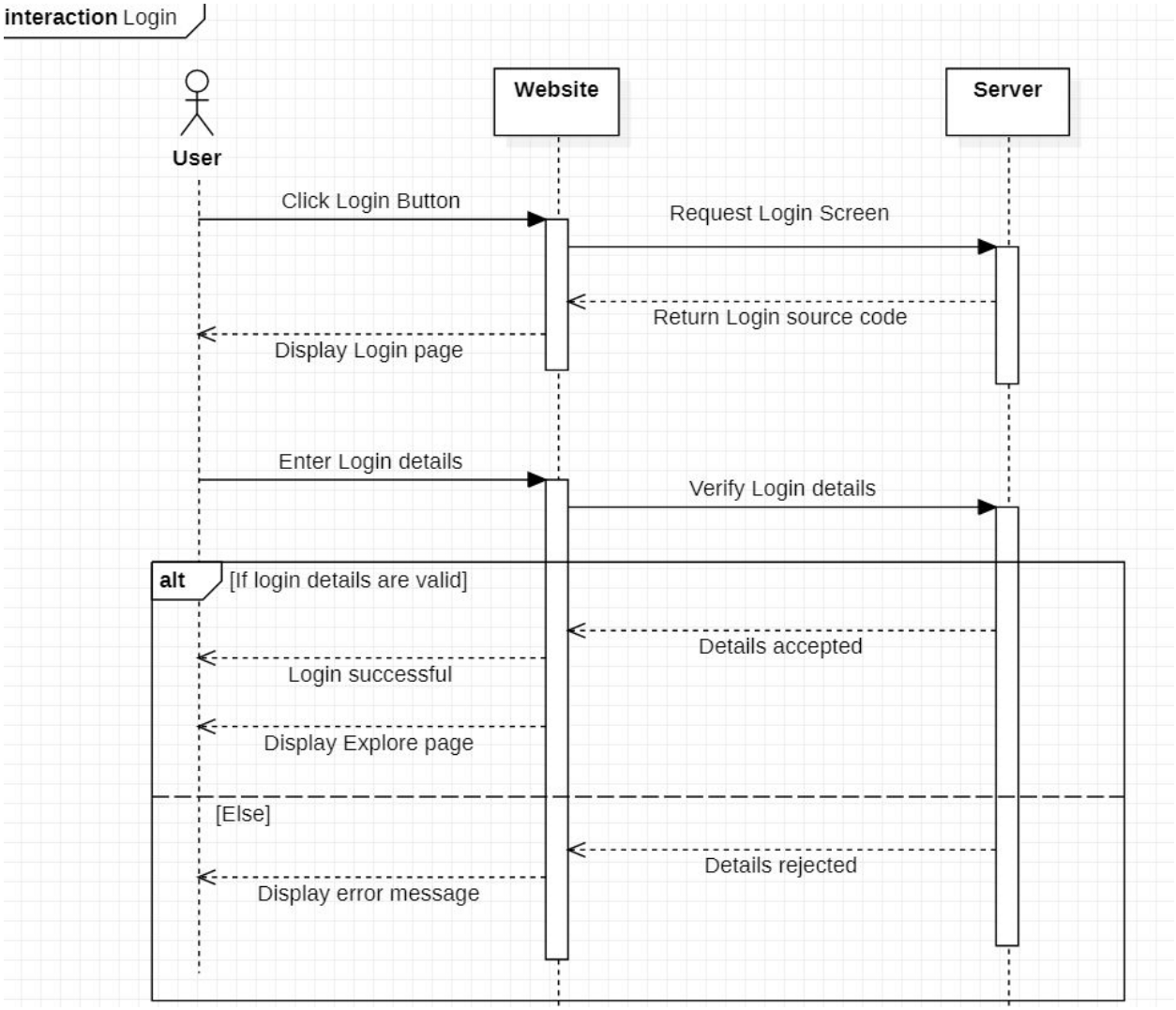
Sequence Diagram 1: User Story 1 - Signup



2. As a student, I want to log into Holler so that I can start using the app.

- Given a user is in the landing page, a ‘Login’ button is available
- Given a user is in the login page, then a user can input an email, password and re-enter password
- When a user logs in with an email, the email must be valid and registered email
- When a user puts in a password, then it should correspond with the email they use to log in with
- When an invalid input is put int, then an error message must be printed
- When I login and the signup has not been fully completed, then users are returned to onboarding
- When I login and registration has been fully completed, then users are sent to the ‘Explore’ tab

Sequence Diagram 2: User Story 2 - Login



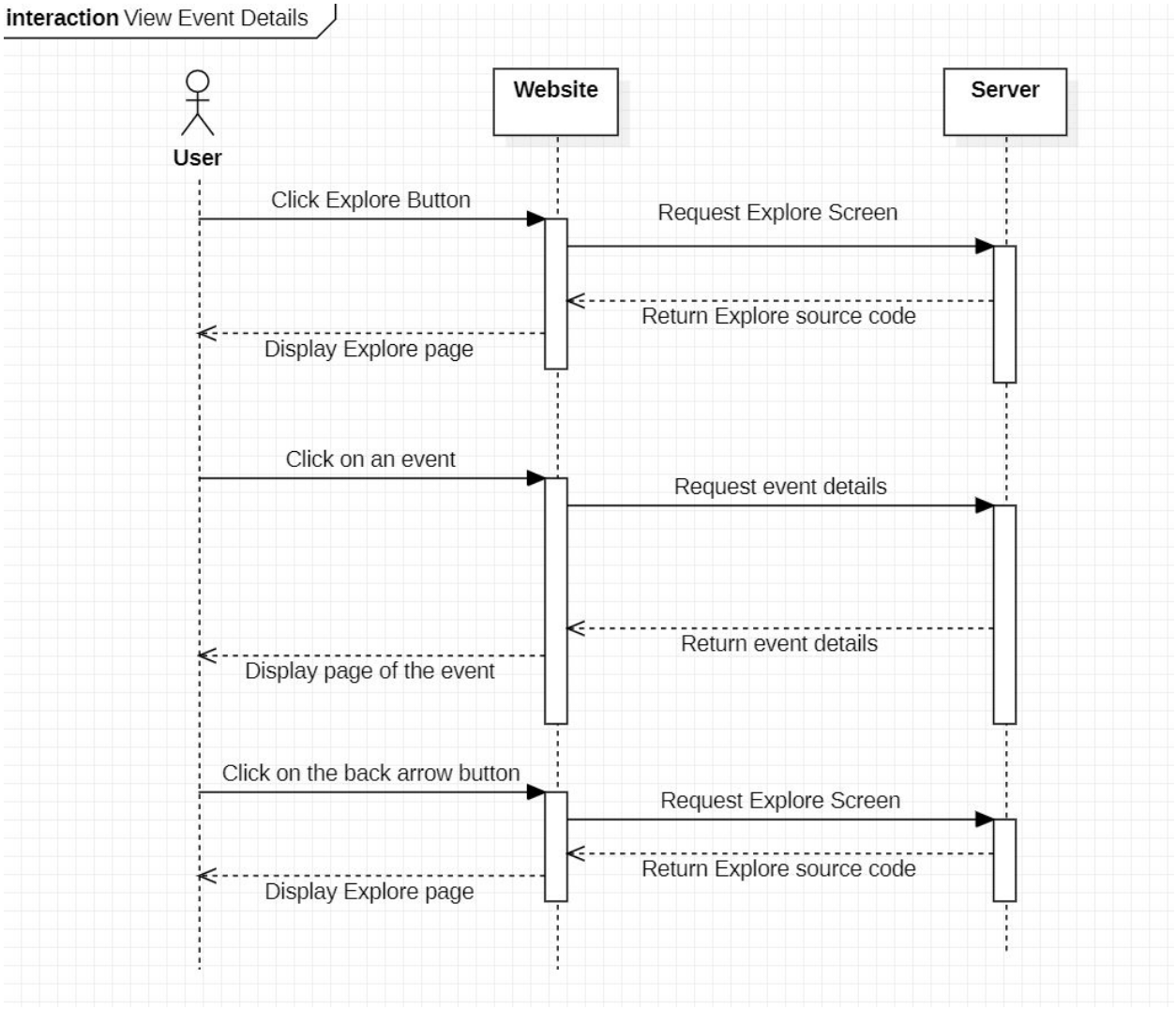
3.2 Events

Epic 2: As a student, I want to search for events so that I can indicate if I am interested in the event or not.

3. As a student, I want to look at an event's details so that I can learn more about the event I am interested in.

- Given a user is in the homepage, then an 'Explore' tab is available
- Given a user is in the 'Explore' tab, then users are automatically presented with society events that match their interests
- Given a user is in the 'Explore' tab, then society events are displayed in a list
- Given a user is in the 'Explore' tab, users are shown an image of the event/society, name of the event, date of the event, location of the event, the tags of the event and how many people are going.
- Given a user is in the 'Explore' tab, then a '+' button is on the bottom right of each event
- Given a user is in the 'Explore' tab, when they click on the event, then it brings up a description of the event and all prior details

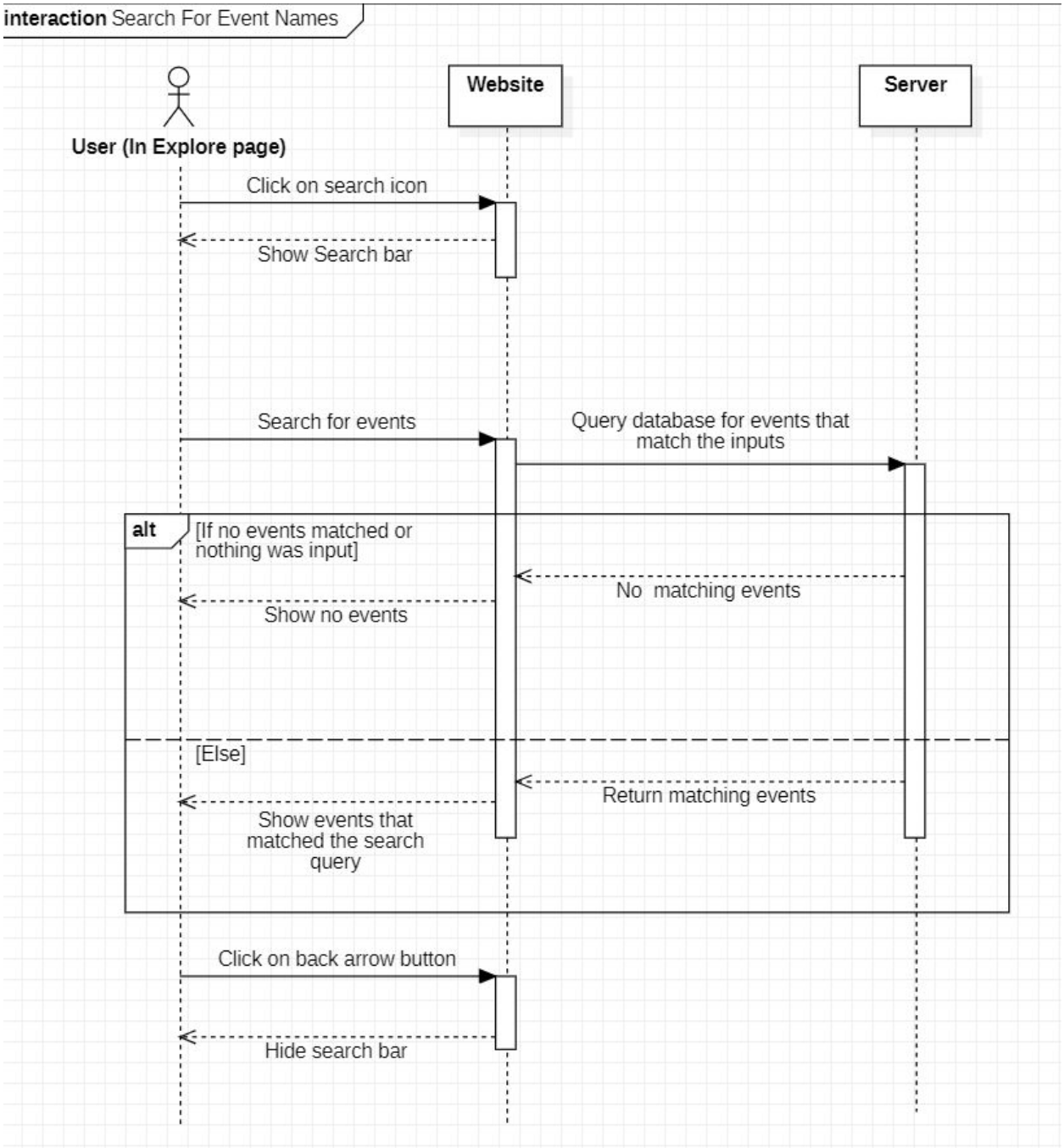
Sequence Diagram 3: User Story 3 - View Event Details



4. As a student, I want to search for events by name so that I can find specific events that I remember being interested in.

- Given a user is in the ‘Explore’ tab, then a searching icon is available near the top of the screen
- Given a user is in the ‘Explore’ tab, when they click on the search icon, then the search bar is displayed and users can type in the search bar for event names.
- Given a user is in the ‘Explore’ tab, when a user is searching for events, then they can click on the back arrow icon to close the search bar.

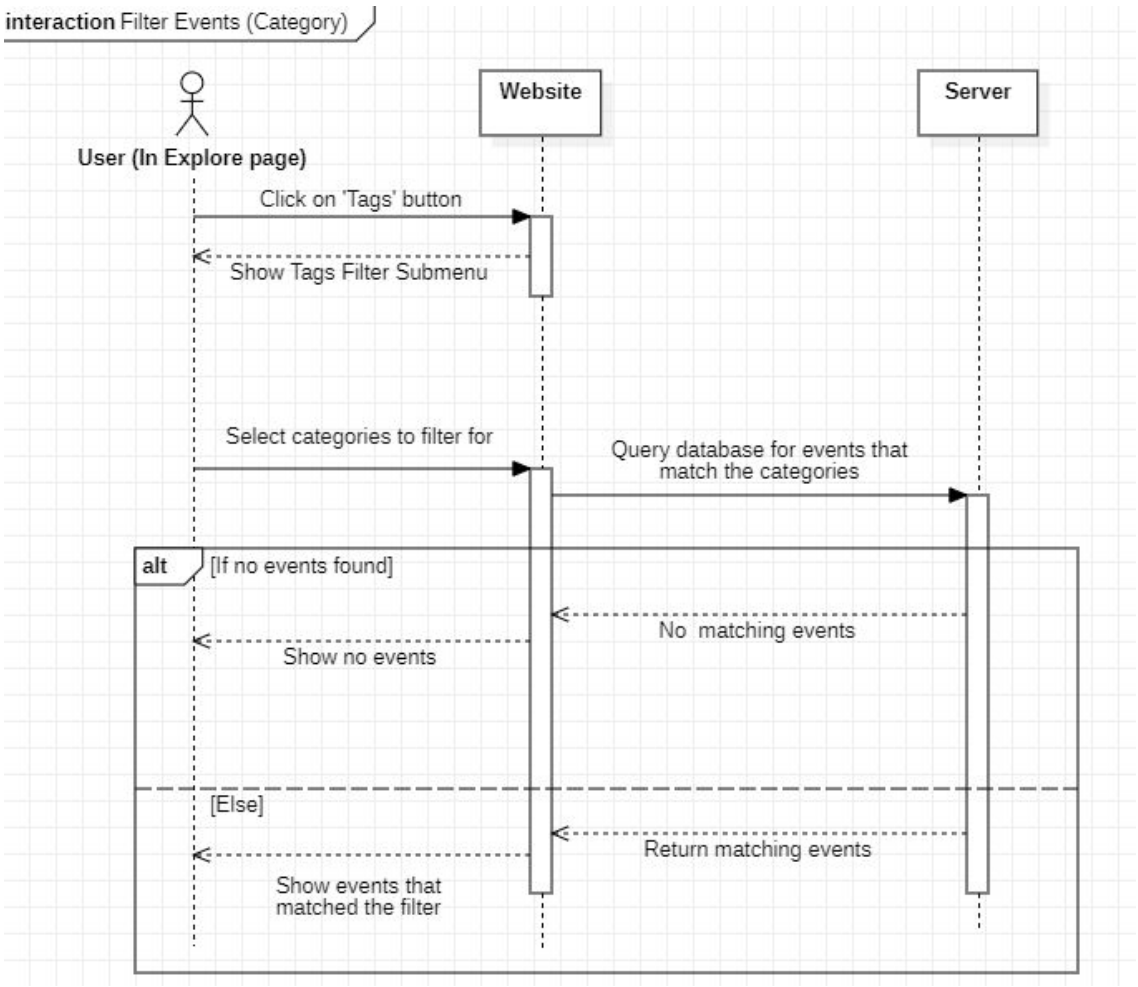
Sequence Diagram 4: User Story 4 - Searching For Event Names



5. As a student, I want to filter events by category so I can find events that match my interests.

- Given a user is in the ‘Explore’ tab, when users click on ‘tags’ users can filter events by tags
- When a user is filtering by tags, then events can be filtered based on interests (e.g. Language, Music, Dancing, Games)
- When a user is filtering by tags, then events can be filtered by location (e.g. UNSW society event or outside event)
- Given a user is in the ‘Filter’ menu, when a user selects a category and searches, then events that match those tags will be displayed
- Given a user is in the ‘Filter’ menu, when a user selects no categories and searches, then it will have no effect

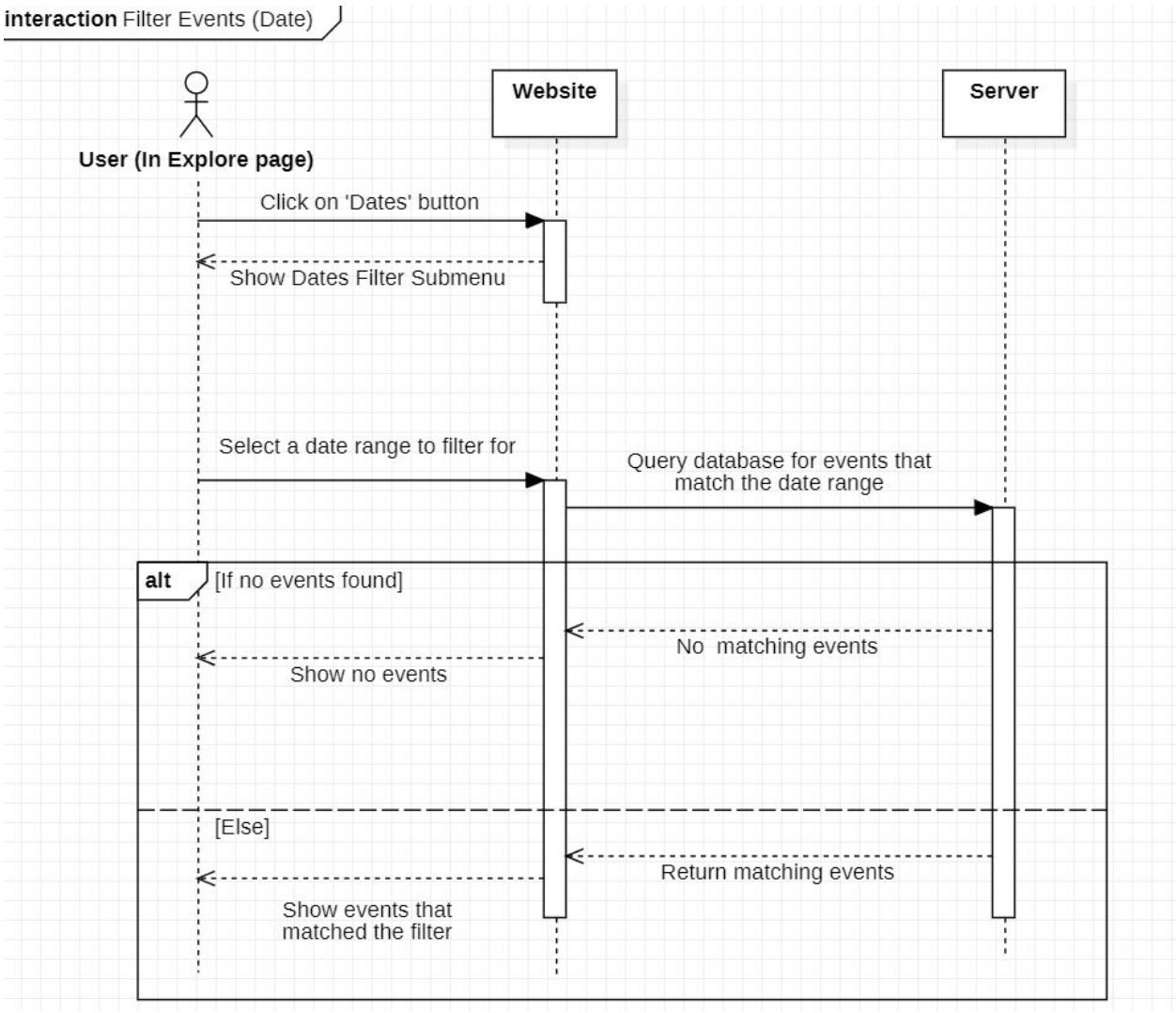
Sequence Diagram 5: User Story 5 - Filtering Categories



6. As a student, I want to filter events by time so I can find events that match my schedule.

- Given a user is in the ‘Filter’ menu, then a user can select multiple categories at a time
- Given a user is in the ‘Explore’ tab, when users click on ‘date’ users can filter events by a time range
- Given a user is in the ‘date’ menu, when a user selects the start date and end date, then the start date must be less than the end date

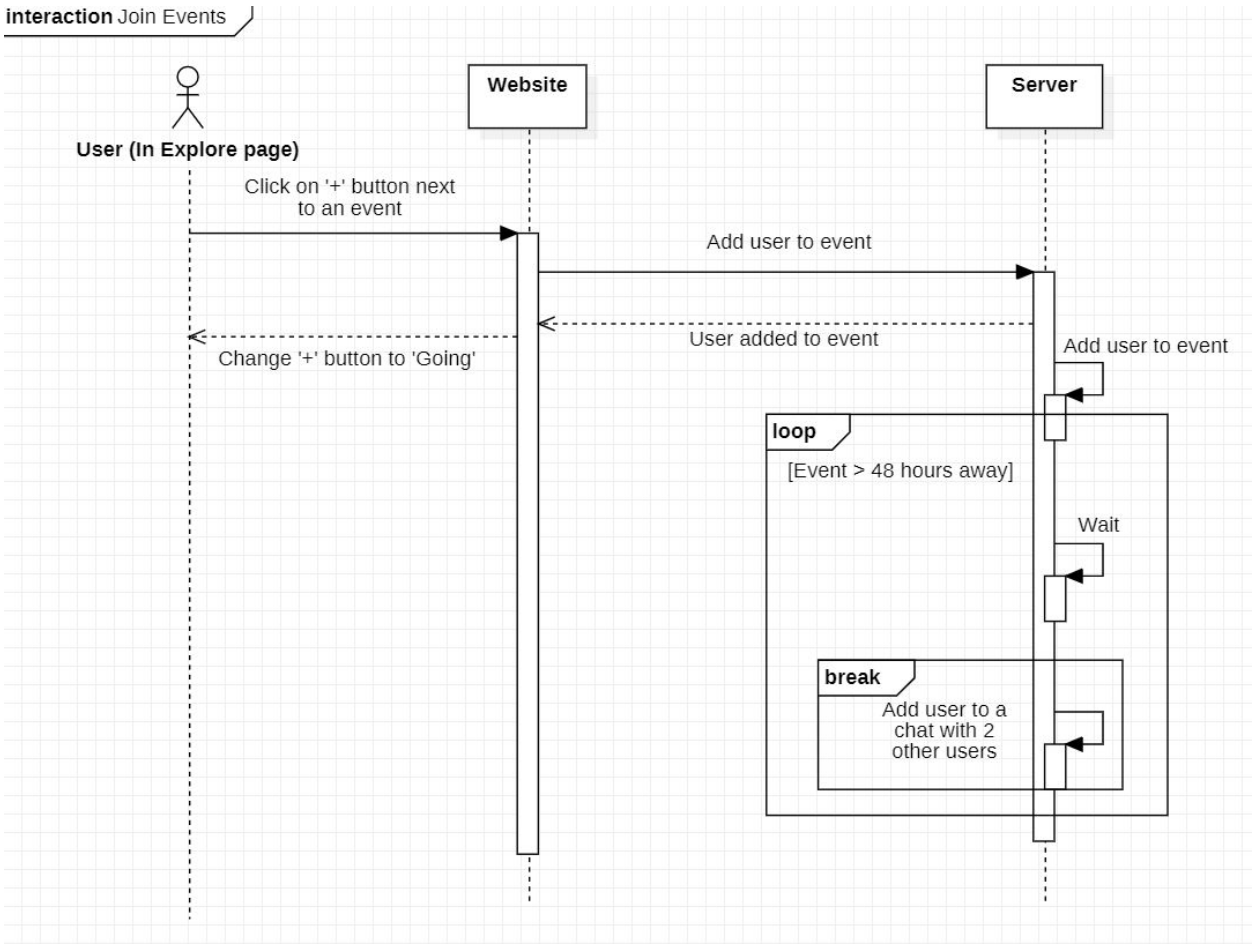
Sequence Diagram 6: User Story 6 - Filtering Dates



7. As a student, I want to join events I am interested in so that I can start talking to people.

- Given a user is in the ‘Explore’ tab, when a user clicks on the ‘+’ button, then a user will indicate they are interested in that society event.
- Given a user is in the ‘Explore’ tab, when the ‘+’ button is pressed, then it changes to ‘Going’ with a tick.
- When a user is going to a society event, then they are matched with 2 other users who are also going to the same event.

Sequence Diagram 7: User Story 7, 8, 9 - Joining an event and getting matched



3.3 Matches / Chat

Epic 3: As a student, I want to be matched with users going to similar events and who have similar interests to me, so that I can meet like-minded people.

8. As a student, I want to be matched with users attending similar events so that I can avoid going alone.

- When a user indicates they are going to a society event, then they are matched with 2 other users who are also going to the same event.
- When a user indicates they are going to a society event, then users are matched 48 hours before an event starts
- Given that a user is in the 'Groups' page, when they are waiting to be matched, then the event is shown in the matches screen with a note labelled "Matching with other users…"
- Given that a user is in the 'Groups' page, when they are waiting to be matched and it is more than 48 hours before the event, then a timer is put next to the event and after it is 48 hours before the event, then they are matched with 2 other users with similar interests.
- When a user is finished being matched, then the matched users are put into a chat

Refer to sequence diagram 7 above.

9. As a student, I want to be matched with user with similar interests so that I can meet like-minded people

- When a user is being matched, then users are matched based on interests, age and classes
- When a user is being matched, then users are matched with the most similar users
- When a user is being matched, and users have matched before for that event, then they cannot match again.
- When a user joins an event, and the user joins within 48 hours before the event, then they are matched with 2 random people, regardless of matching interests.

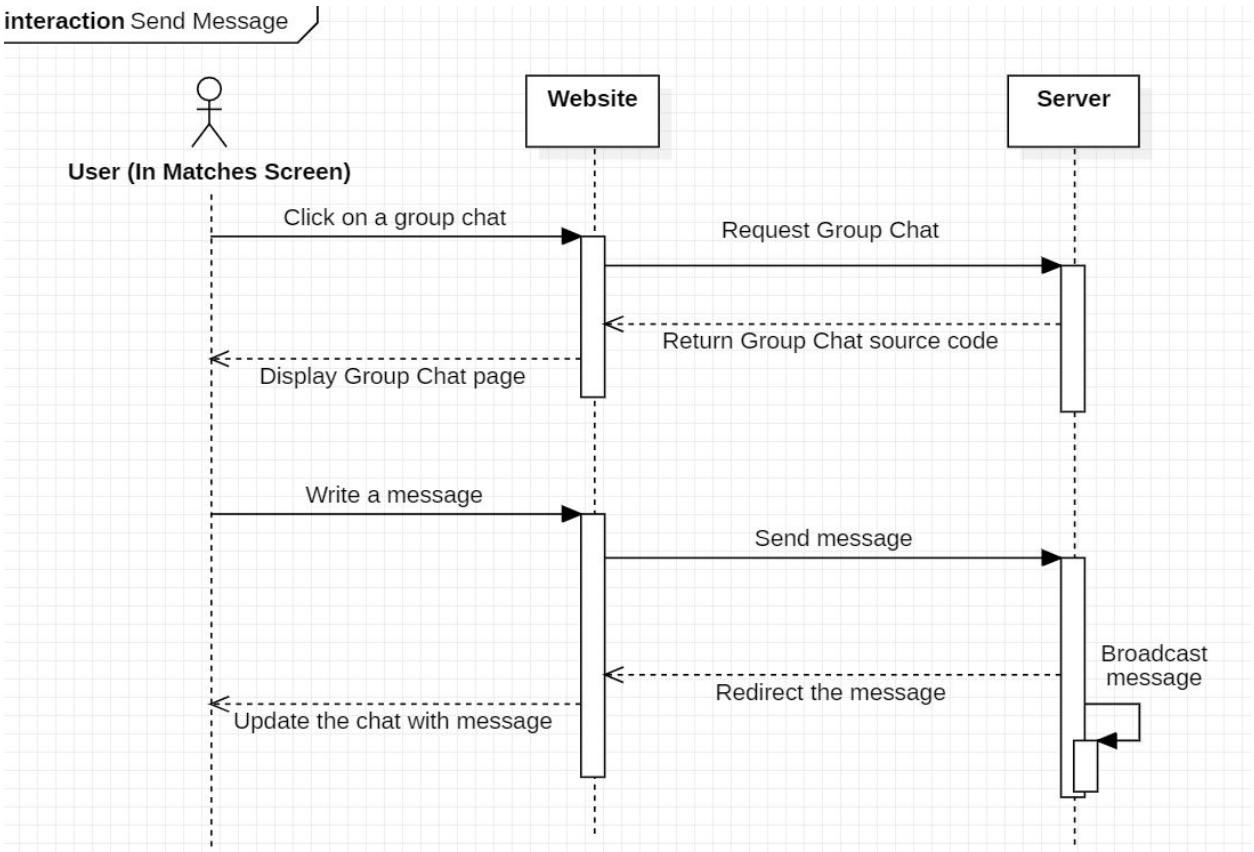
Refer to sequence diagram 7 above.

Epic 4: As a student, I want to chat with multiple people so that I can get to know more people.

10. As a student, I want to send messages so that I can communicate with other users.

- When a user is matched, then a chat is created with 2 other users
- Given that a user is in the home page and they are logged in, then a ‘Groups’ tab is available in the navigation menu.
- Given that a user is not currently in the ‘Groups’ page, then clicking the tab brings up a list of all current chats.
- Given that a user is in the ‘Groups’ page, then a user should see each group chat with: the first names of the users in the chat, the most recent message, who sent the most recent message and when it was sent, and which event the group is going to.
- Given that a user is in the ‘Groups’ page, when a user clicks on the group chat, then it opens up the chat
- Given a user is in an opened group chat, then users can send text messages
- Given a user is in an opened group chat, then past messages are saved and can be viewed later

Sequence Diagram 8: User Story 10 - Sending a Message



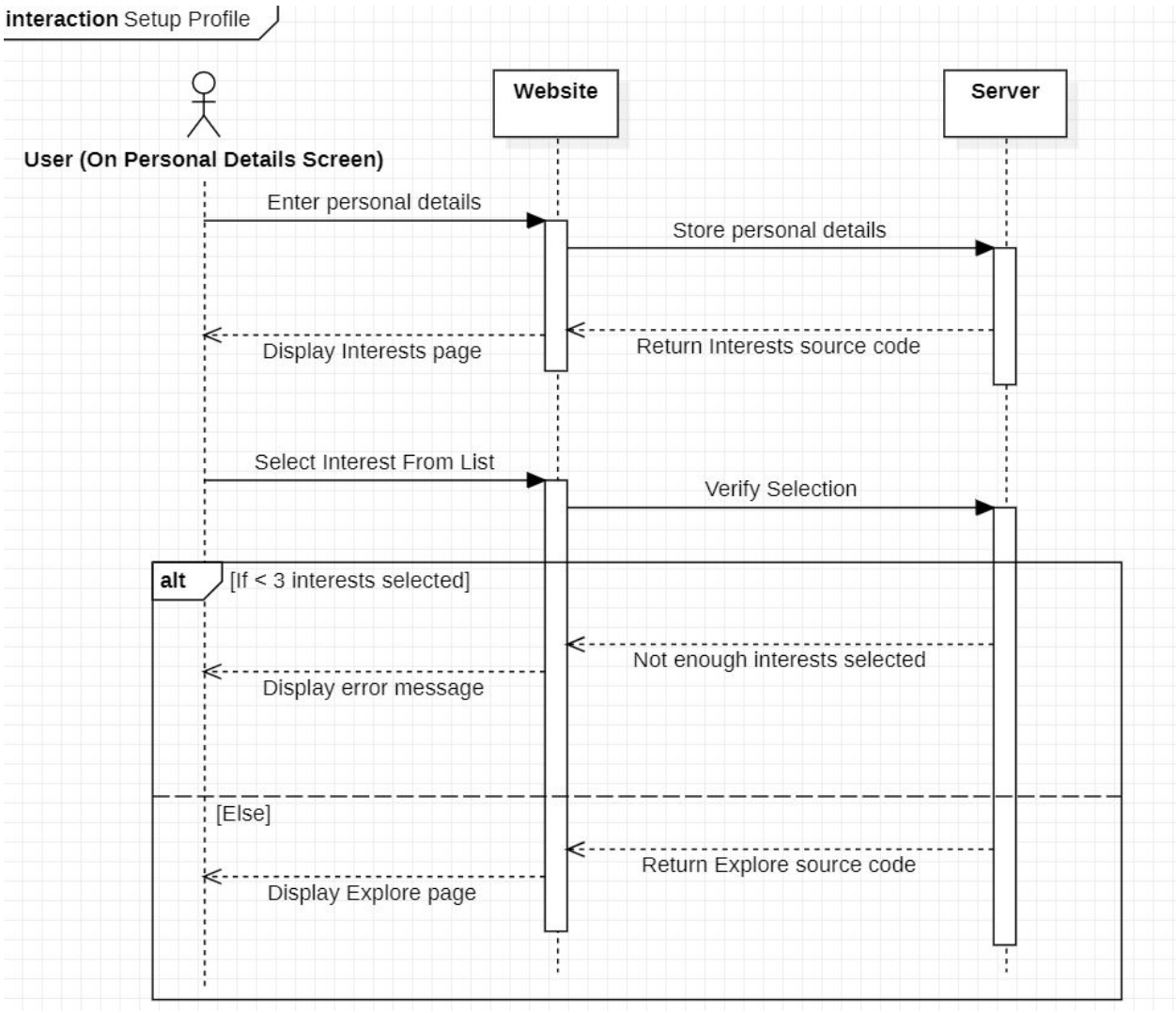
3.4 Profile

Epic 5: As a student, I want to view my profile and change my settings so that all my details are accurate, and I can keep track of my progress.

11. As a student, I want to set my personal details so that I can be matched according to my preference.

- When a user is signing up and logging in for the first time, then users can set their details
- Given a user is in the onboarding section, then users must put their first name
- Given a user is in the onboarding section, then users must put their date of birth
- Given a user is in the onboarding section, then users must select their faculty
- Given a user is in the onboarding section, then users must select their university (UNSW by default)
- Given a user is in the interests section of the onboarding process, then users can select from a list of various interests
- Given a user is in the interests section of the onboarding process, then a user must select at least 5 interests
- Given a user is in the avatar creation section of the onboarding process, then a user must create an avatar.

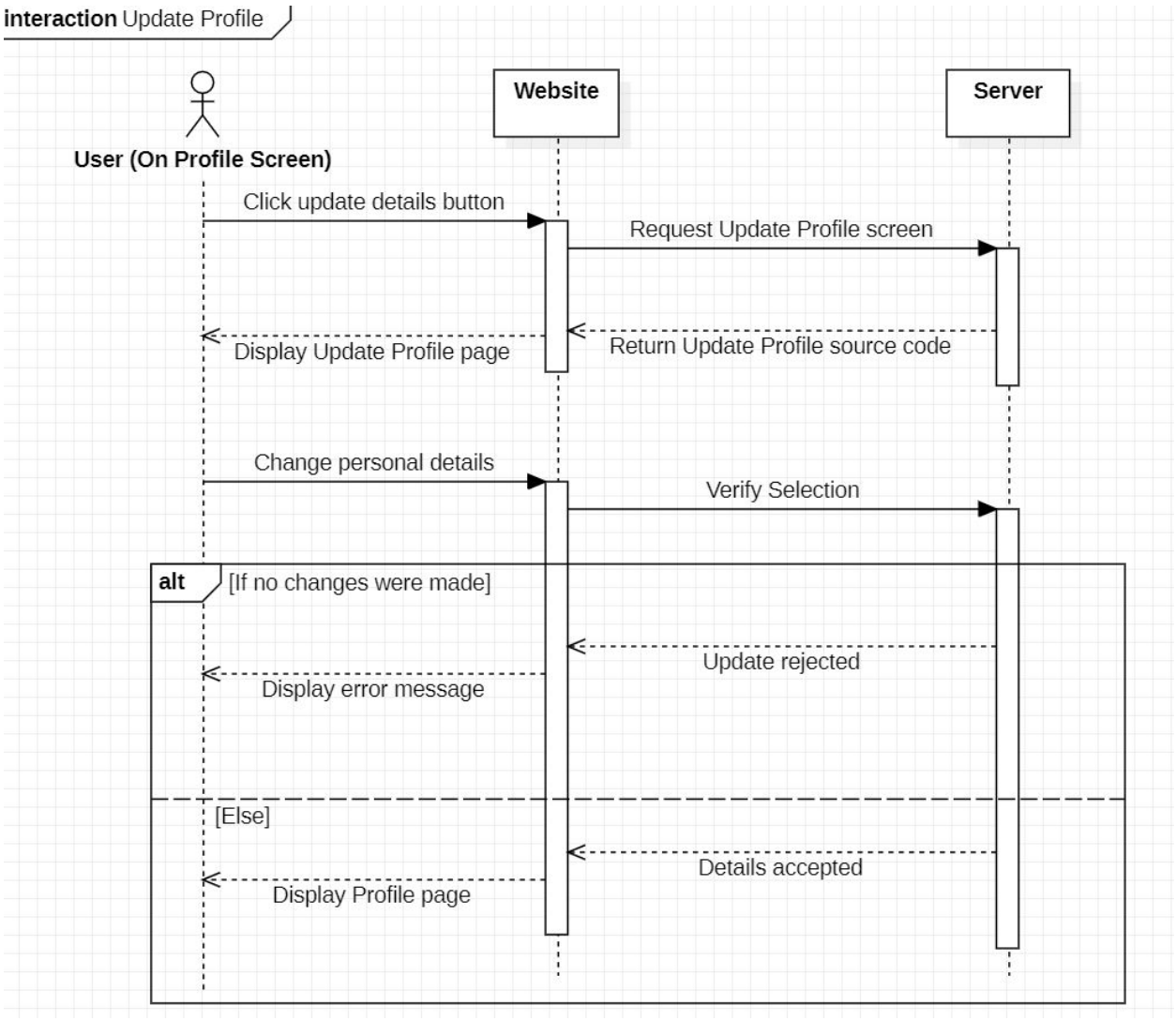
Sequence Diagram 9: User Story 11 - Setup Profile



12. As a student, I want to update my personal details, so that my interests and details are up to date.

- Given a user is in the ‘Profile’ tab, when they click on edit profile, then they should be able to see their profile editing screen
- Given a user is in the profile editing screen, then users can change their list of interests by checking or unchecking from the list.
- Given a user is in the profile editing screen, then users can change their faculty
- Given a user is in the profile editing screen, then users can change their first name
- Given a user is in the profile editing screen, then users can change their date of birth
- Given a user is in the profile editing screen, then users can change their avatar.

Sequence Diagram 10: User Story 12 - Update Profile

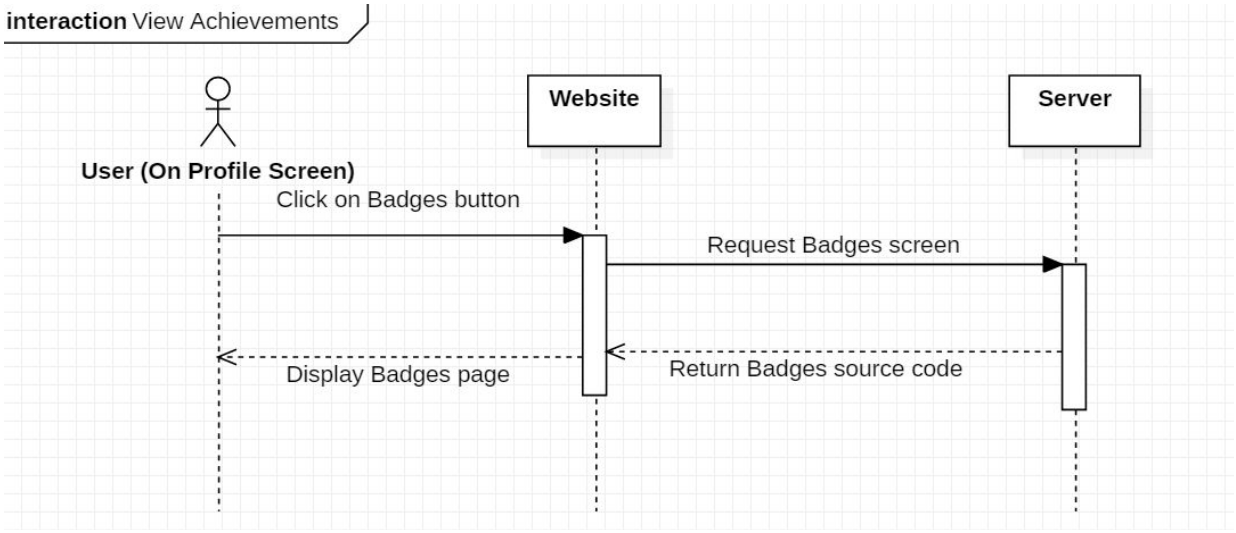


Epic 6: As a student, I want to view my achievements and statistics so I can keep track of how much progress I have made in the app.

13. As a student, I want to keep track of my achievements, so that I can feel a sense of pride and accomplishment.

- When a user completes a certain task for the first eim, then users are given a badge from a list of achievements (Examples: Change your avatar, Match with someone for the first time, Match with 10 people)
- Given a user is in their profile, when they click on ‘Badges’, then users can view the badges they have earned
- When a user earns a badge, then users are given ‘points’

Sequence Diagram 11: User Story 13 & 14 - Viewing Badges



14. As a student, I want to view statistics related to my account, so I can keep track of how much I have used the app.

- Given a user is in their profile, then users can view how many groups they have joined
- Given a user is in their profile, then users can view how many badges they have earned
- Given a user is in their profile, then users are provided with a graph that tracks their score and their percentile rank

Refer to sequence diagram 11 above.