

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

**IM3080 Design and Innovation Project
PROJECT REPORT**

BUSMEET

GROUP 4

Chua Jing Yi | Goh Hongyi, Sebastian | Ho Jia Jun | Huang Bin Melville

Kho Yan Qin | Lee Yi Xuan | Lim Li Swen | Lin Jin Zhao | Loh Jeng

Soo Keng Shuang Serena | Teng Say Gek

Supervisor: Assoc Prof Erry Gunawan

GitHub: https://github.com/lisllin99/IEM-DIP-G4/tree/busmeet_v1

School of Electrical and Electronic Engineering

November 2021

Contents

Background & Motivation	1
1.1 Problem Identification.....	1
1.2 Evaluating Current Application	1
Objective & Solution	2
2.1 BUSMEET.....	2
Review of Literature/Technology.....	3
3.1 Technological Stacks.....	3
3.1.1 Figma: Application Design & Wireframe.....	3
3.1.2 React Native: Application Development.....	3
3.1.3 Amazon Web Services (AWS) Amplify	4
3.1.4 Visual Studio Code	5
3.1.5 Emulators	6
3.2 Technical Platforms	6

3.2.1 PowToon	6
3.2.2 Clip Studio Paint.....	7
3.2.3 GitHub	7
3.2.4 Google Drive	7
3.2.5 Discord	8
Design & Implementation	8
4.1 Getting Started	8
4.1.1 Selecting Features & Components	8
4.1.2 Choice of Programming Language	9
4.1.3 Choosing Backend Services	9
4.1.4 Google APIs	10
4.2 Design	11
4.2.1 Figma/MIRO	11
4.3 Implementation	13
4.3.1 Walkthrough/ Introduction Screen.....	13

4.3.2	Home Screen	14
4.3.3	Selecting Destination, Time, and Inviting Other Users	15
4.3.4	Displaying All Users Location.....	16
4.3.5	Chat.....	18
4.3.6	Checking Bus Information	19
4.3.7	Accepting Meet-Ups and Transferring Buses	20
4.3.8	React Native with AWS Amplify.....	21
4.3.9	Character Avatars	23
4.3.10	Application Design	24
4.3.11	Use Cases.....	26
4.4	Challenges	27
4.4.1	Lack of Experience	27
4.4.2	Communication	27
4.4.3	Time Restriction.....	28
4.4.4	Integration of Features to main application.....	28

Conclusion & Recommendation	29
5.1 Conclusion	29
5.2 Recommendation for future works.....	30
Appendices	31

Chapter 1

Background & Motivation

1.1 Problem Identification

Initially, our group wanted to focus on creating an application that could be an improvement to existing bus applications that were used to check bus information. From our personal experiences, we noticed that when we wanted to travel with our friends via buses to our destination, we often used applications such as WhatsApp or Telegram to share our location with our friends. At the same time, we would also have to keep track of when the bus is coming by switching between applications. Thus, we felt that more could be done to integrate these features. From here, we came up with the goal of our project, which was to create an integrated application where users can check bus information while interacting with other users, such as messaging, location sharing, and sharing their estimated time of arrival (ETA).

1.2 Evaluating Current Application

With the initial problem identification, we decided to work on the application by first looking at existing applications and the features we could make use of. We took note of two apps, which were Glympse and BusLeh. To create an improved version of a bus application we felt that features of these two applications could be integrated together. Additionally, the current Glympse application only allows users

to send a short message to another user instead of being able to communicate live with each other. Furthermore, with no public transport being listed in Glympse, we decided to narrow down our target audience to Singapore students with the goal of incorporating buses into the app as a form of transport as taking public transport is such a large part of a student's life.

Chapter 2

Objective & Solution

2.1 BUSMEET

With the above problem mentioned, our group decided to work on an integrated application with this objective:

“Creating an all-in-one application where users can check bus information while interacting with other users, such as messaging, location sharing, and sharing their estimated time of arrival (ETA).”

Our application will be mainly used for users to travel with their friends or other users they want to meet up with during their journey there while they are taking buses as their main mode of transport. Our application will serve as a more convenient way for users to locate their friends or other users when they have reached their destination as well.

Chapter 3

Review of Literature/Technology

3.1 Technological Stacks

3.1.1 Figma: Application Design & Wireframe

Figma is a vector graphics editor and prototyping tool that can create designs for both Android and IOS. It is a good tool to prototype the different iterations of the app from the different perspectives of our group members. Additionally, collaboration and sharing of design ideas were also easy as we could all edit in real-time.

3.1.2 React Native: Application Development

React Native is an open-source UI software framework created by Facebook, Inc. We chose React Native as our language of choice as it has a big community where many questions are answered on sites like Stack Overflow. So, if we were to be faced with any difficulties, we would be able to look up solutions that are available online. React Native also supports a "Code once, run anywhere" goal, which means by creating our application with it, we can run it on any platform like Android or IOS.

3.1.3 Amazon Web Services (AWS) Amplify

AWS Amplify is a set of purpose-built tools and services that makes it quick and easy to set up the backend infrastructure of our mobile app without the need of writing long lines of code. Tools we utilized on AWS amplify were **Amazon Cognito, AWS Lambda, Appsync, DynamoDB and GraphQL**

The following are how we made use of these tools and services:

Amazon Cognito:

Authentication and authorization of user into the app (logging in and out)

AWS Lambda:

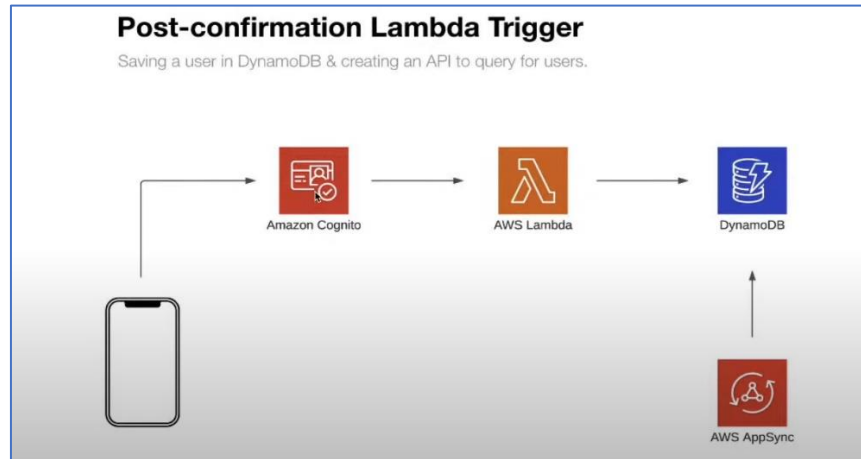
Serverless computing platform that allows us to run backend code without managing any servers

Appsync and GraphQL:

Retrieve/Store the live location of users from **DynamoDB**

DynamoDB:

Store user information (location, email, username, password, etc.)



Post-Confirmation Lambda Trigger-AWS

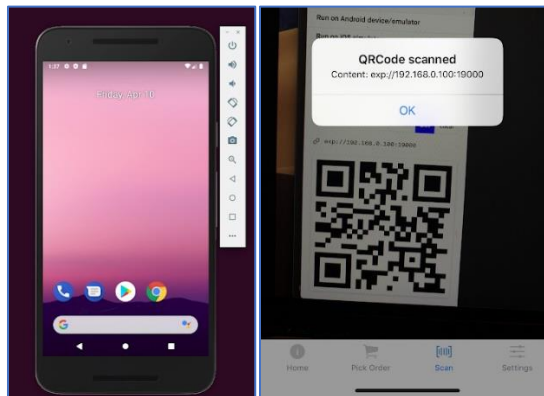
3.1.4 Visual Studio Code

Visual Studio Code (VS Code) is a free source-code editor made by Microsoft for Windows, Linux, and macOS. VS Code was our main Integrated Development Environment (IDE) of choice due to its convenient features and shortcuts, as well as having accessible React extensions that make development more convenient. Pushing code from VS Code to GitHub was also less of a difficulty as they have built-in extensions and features that allow code to be pushed onto GitHub in seconds.

3.1.5 Emulators

Since our product is going to be cross-platform, we decided to run our app on emulators. We have chosen 2 emulators, Android Studio emulator for testing the app on Android system and using iPhone to test it on IOS system.

Even though we cannot use Android Studio for building our app, we can still make use of the emulator that Android Studio provided. It is a tool that everyone has experience with and there is no need to spend time finding another emulator.



Android Studio Emulator & Physical device

3.2 Technical Platforms

3.2.1 PowToon

PowToon is an e-Tool that creates animated videos for personal, educational, or business/professional use. PowToon allowed us to create an interactive,

engaging, and informational video about our application, the creation, and editing of video were smooth, and exporting of video was hassle-free.

3.2.2 Clip Studio Paint

Clip Studio Paint is a software for digital drawing and painting created by CELSYS, Inc. As one of our members already had the program and was familiar with it, we used it as our main software for drawing custom assets.

3.2.3 GitHub

GitHub provides hosting for software development and version control using Git. It offers distributed version control and source code management which are crucial in our collaboration efforts. GitHub allowed us to use a shared repository where we could merge codes made by each other, making collaboration easy. Retrieving previous versions of the code was easy as well.

3.2.4 Google Drive

We used Google Drive (Google Docs and Google Slides) as our main platform to share important information and resources such as ideation documents, meeting minutes, tasks, and actions, as well as work on reports and presentation slides in real-time.

3.2.5 Discord

Discord is a communication platform where users can communicate with voice calls, video calls, text messaging, media, and files in private chats or as part of communities called "servers". Using discord, we could share code and ideas easily through screen sharing. Individual teams (e.g., UI/UX team and development team) could also go into their private servers to discuss any ideas they have without interfering with each other's discussion.

Chapter 4

Design & Implementation

4.1 Getting Started

4.1.1 Selecting Features & Components

We started to brainstorm our ideas over the course of different weeks to make our application unique and creative. We chose features that were implemented before and that we could recreate with our improvements in mind. The main selling point of our application is the convenience factor for the user whereby a user can know where his or her friend is at, what bus they are on, and their estimated arrival time while also being able to check when their bus is arriving as well.

4.1.2 Choice of Programming Language

Our main objective for the language was to allow us to create an application that could run on multiple platforms (e.g., IOS and Android).

After comparing different languages, we chose React Native which has multiple advantages. This includes a large community where many of the common bugs or issues faced by developers were answered on sites like Stack Overflow. Furthermore, React Native allows for reusable components. For example, the same code used to create a button could be used on multiple pages without the need to rewrite the code, which saved us lots of time when trying to implement our application.

To write our code in React Native, we decided to go with the Expo framework, which is an open-source platform for making universal native apps for Android, iOS, and the web with JavaScript and React. Expo allows us to write React Native code without additional dependencies for native IOS or Android code.

4.1.3 Choosing Backend Services

The next step was to choose a secure and reliable backend service that could allow us to transmit real-time information from the server. With the limited amount of time we had to complete our project, we understood that utilizing a third-party service could help us tremendously by saving a lot of time and effort to write backend code.

The tools and services we used are explained in Section 3.1 and the reason why we chose to go with AWS Amplify is its ease of use. A single user interface known as the amplify console allows us to write very small amounts of code to retrieve and store data using its myriad of services. Furthermore, we do not have to worry about testing our backend code to check whether it is working when running our application.

4.1.4 Google APIs

Google APIs were utilized for our application to display maps with additional features, as well as search for nearby locations. The primary services used were the Google Maps API as well as the Google Geolocation API.

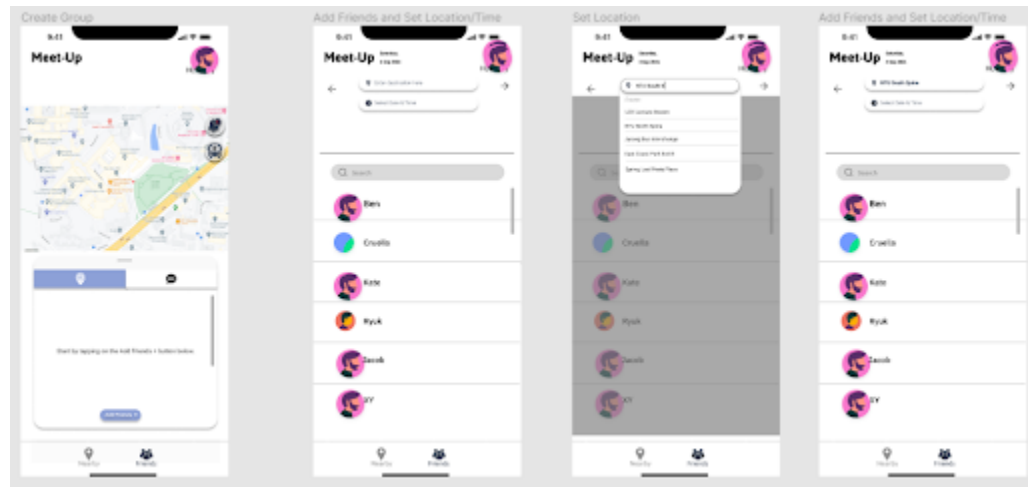
Places API: Used to display places that are defined within API as establishments, geographic locations, or prominent points of interest. We utilized this API to display these places on the map when searched.

Directions API: Used to display directions from starting location to destination.

Geolocation API: Used to obtain a location and accuracy radius based on information about cell towers and Wi-Fi nodes that the mobile client can detect.

4.2 Design

4.2.1 Figma/MIRO



Figma Design Page

We started the initial design and prototype process using Figma to convert all our brainstorm ideas into screens. Through Figma, we planned out the initial stages and functions of the app we wanted to include. Next, we did a trial run on Figma to have a glance at what our final app looks like.



MIRO Whiteboard

With MIRO Whiteboard features, we imported the screen designs from Figma. From there, we linked up the screens with arrows to create a flow diagram as shown above. This allows us to be able to visualize better whenever a button is clicked or when the user does something, to bring them to the next screen.

4.3 Implementation

4.3.1 Walkthrough/ Introduction Screen

The user will first land on a walkthrough page where they will be met with three different types of splash screens. These splash screens will guide the user on the different features available on our application itself. Users will slide across the different splash screens (Figure 1 -> Figure 2 -> Figure 3) and after clicking on the Done button, they will be directed to the home screen (Refer to section 4.3.2). Users can also click on the Skip button if they wish to skip the entire walkthrough.

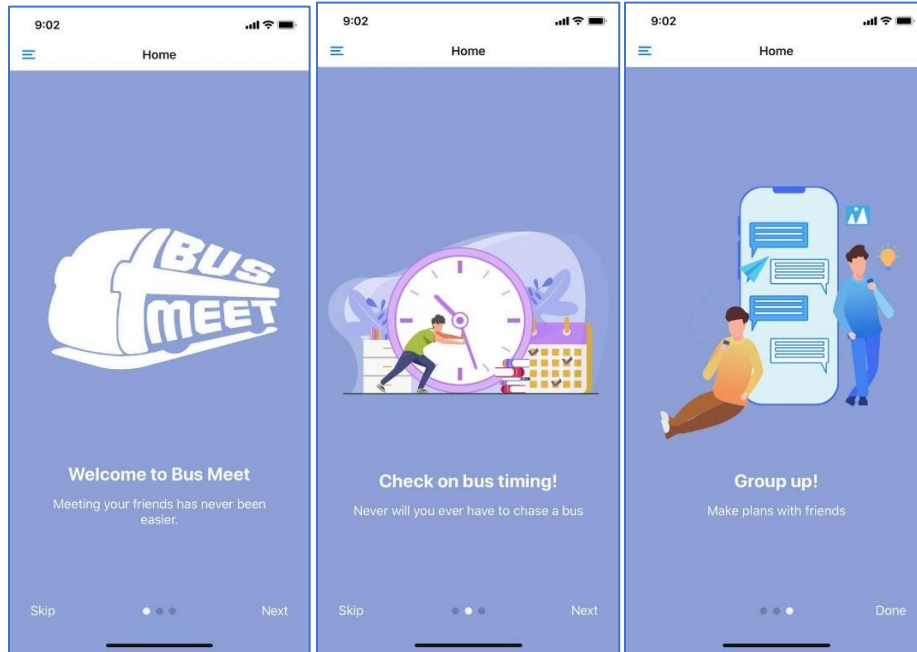


Figure 1

Figure 2

Figure 3

4.3.2 Home Screen

At the home screen (Figure 4), users will be able to see their current location and by clicking on the Meet Where area or Add Friends button, they will be brought over to the next screen, where they could input the necessary information for the meetup and invite other users. The real-time location display, as well as map display, is done with the help of the Google Maps API.

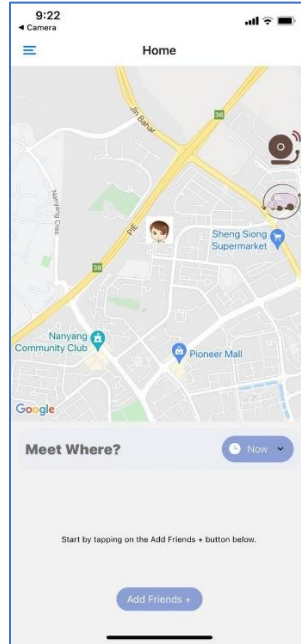


Figure 4

4.3.3 Selecting Destination, Time, and Inviting Other Users

The search function is implemented (Figure 5) using the Google Maps Autocomplete API whereby nearby locations would be displayed on the search results. After users have inputted all the necessary information and selected all the users they want to invite (Figure 6), they will be automatically moved to the next screen, where it will display all the users' locations and estimated arrival time once they accept the invitation to the meet-up.

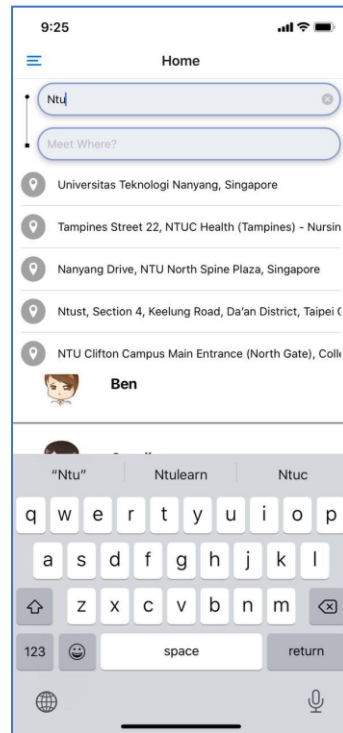


Figure 5

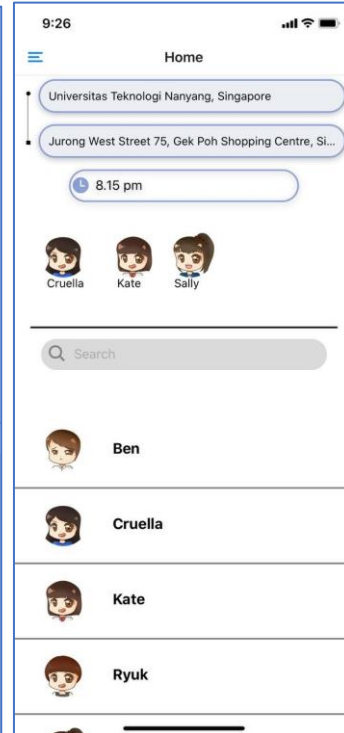


Figure 6

4.3.4 Displaying All Users Location

All the users' locations and their estimated arrival time will be displayed on the map, where their live locations are displayed (Figure 7) by retrieving them from the DynamoDB database with the use of Appsync and GraphQL.

Once the user has reached their destination, there will be an alert that pops out to notify them of their arrival. (Figure 8).

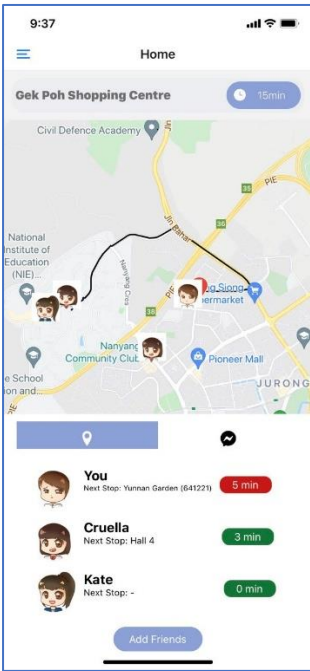


Figure 7

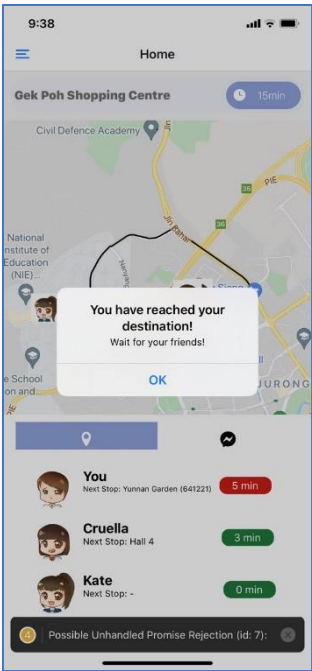


Figure 8

4.3.5 Chat

Users will also be able to communicate with other users via the chat function (Figure 9) which helps to ease any difficulties faced if they are unable to find each other.

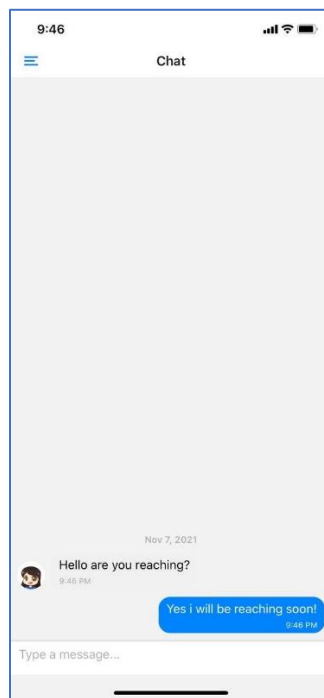


Figure 9

4.3.6 Checking Bus Information

Users will also be able to check bus information like estimated arrival timings, buses, and first and last bus timings.

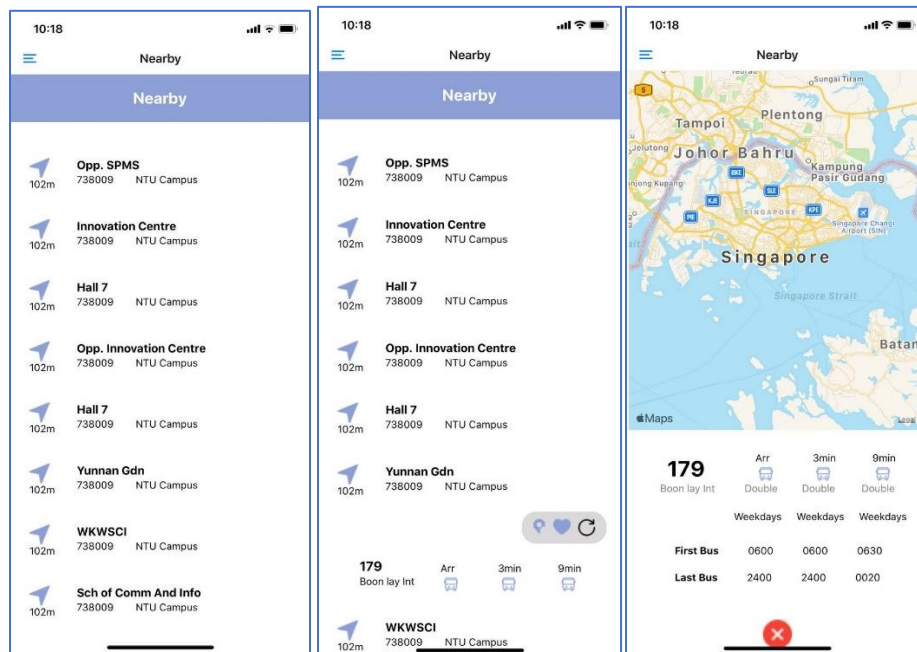


Figure 10

Figure 11

Figure 12

4.3.7 Accepting Meetups and Transferring Buses

Users will also be able to accept meet-up invitations and choose the bus they are going to board at the same time (*Figure 13*). They also have the option to transfer buses if they are switching buses during their journey (*Figure 14*). Both the Board and Transfer buttons will bring users to the destination search page, where they could update their locations again.

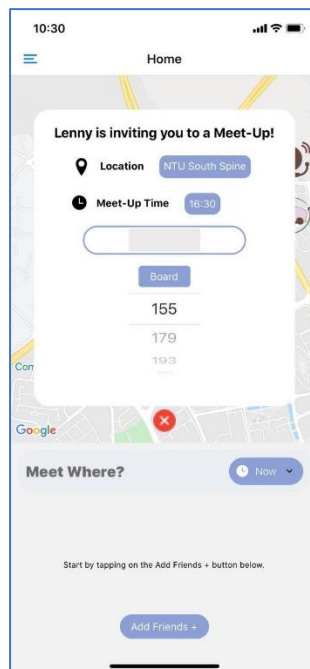


Figure 13

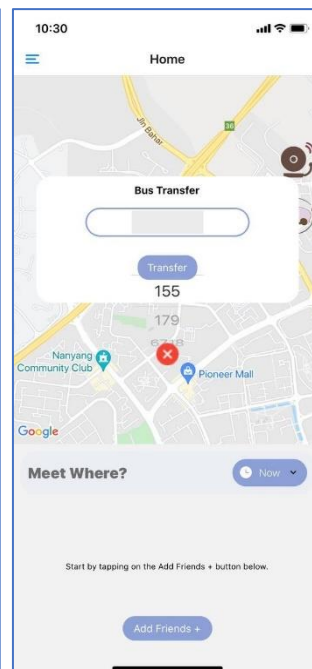


Figure 14

4.3.8 React Native with AWS Amplify

A few features were integrated using tools and services from AWS Amplify. We first implemented the login and log-out feature (*Figure 15*). This was done with the help of AWS Cognito which is a service provided by AWS (Amazon Web Services) for authentication and authorization of the user. When a user signs up using the interface provided (*Figure 16*), their information will be automatically stored in the DynamoDB database (*Figure 17*).

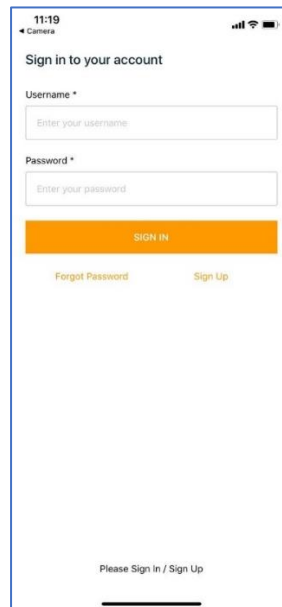
A mobile app interface for logging in. At the top, it says "Sign in to your account". Below this are two input fields: "Username *" with a placeholder "Enter your username:" and "Password *" with a placeholder "Enter your password:". Below the password field is an orange "SIGN IN" button. Underneath the button are two links: "Forgot Password" and "Sign Up". At the bottom, there is a link that says "Please Sign In / Sign Up". The status bar at the top shows the time as 11:19 and a camera icon.

Figure 15

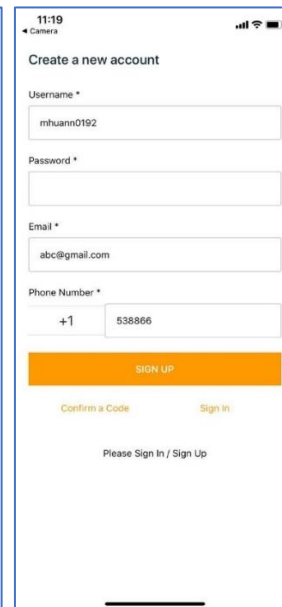
A mobile app interface for creating a new account. At the top, it says "Create a new account". Below this are four input fields: "Username *" with the value "mhuan0192", "Password *" (empty), "Email *" with the value "abc@gmail.com", and "Phone Number *" with a dropdown showing "+1" and a text input with "538866". Below these fields is an orange "SIGN UP" button. Underneath the button are two links: "Confirm a Code" and "Sign In". Below these links is a link that says "Please Sign In / Sign Up". The status bar at the top shows the time as 11:19 and a camera icon.

Figure 16

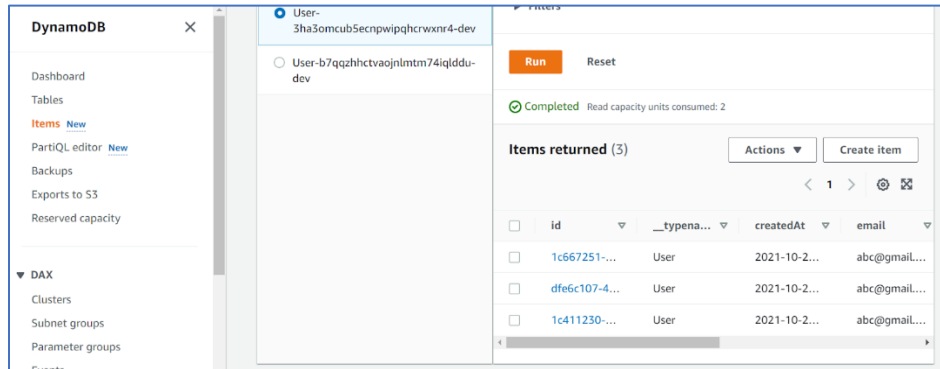


Figure 17

Furthermore, the location of users (latitude and longitude) is retrieved from the DynamoDB database using AppSync and GraphQL, Appsync utilizes GraphQL to store and retrieve information from the DynamoDB database. We are also able to easily query the data using just a few lines of code in the Queries section of AppSync (Figure 18).

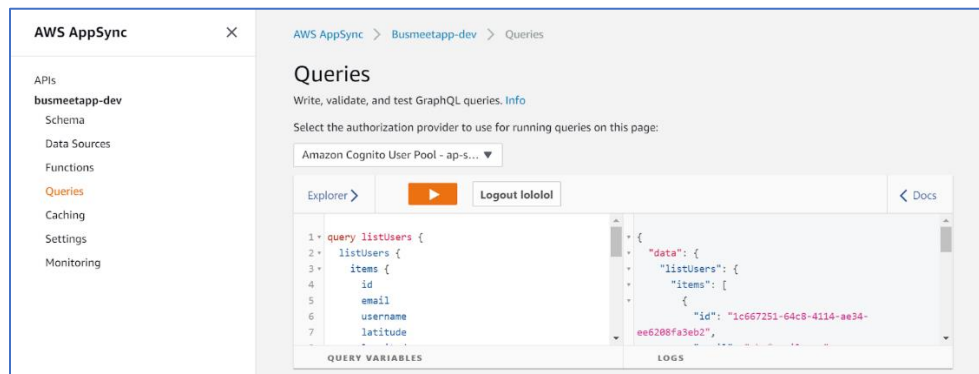


Figure 18

4.3.9 Character Avatars

Users may use it to represent themselves in the application. This allows users to customize their avatars to either look like themselves or what they want to use as a representation. Furthermore, users can identify their friends easily with the different avatars designed by their friends, making it easier to spot their friends' location on the map immediately.

Even though we were unable to integrate the actual functionality of creating the customizable avatars into the current application due to the time limitations, we have designed the interface and process of how this will look in the application.

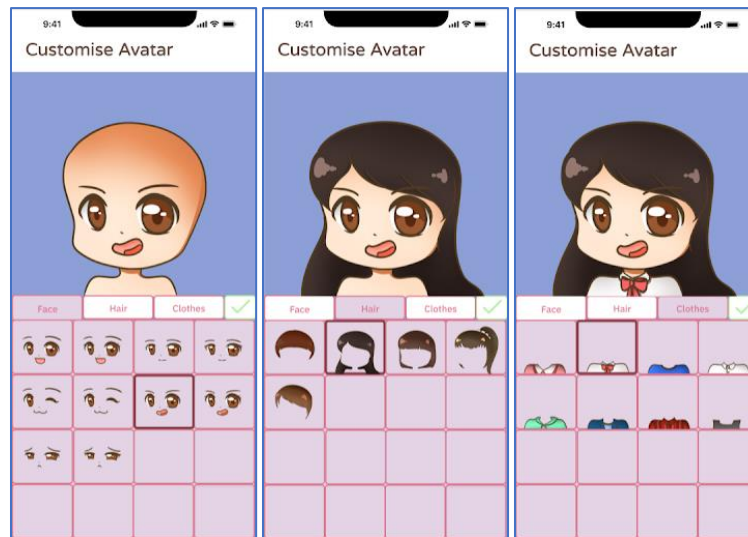


Figure 19

Figure 20

Figure 21

We created 150 permutations of different avatars to choose from. These permutations are limited through specifying faces with prominent eyelashes, longer hairstyles, and certain clothing items to be for female avatars and the rest to be for male avatars.

4.3.10 Application Design

Overall, our user interface team settled on a school-like theme for the application as our target audience likely fell in the category of students, be it high school or university students. Even for working adults, the theme would likely be able to bring in nostalgia and would not exclude them, creating a more friendly feeling application. Hence, we also designed the customizable avatars to include some clothing resembling school uniforms and designed a teacher mascot (Figure 22) for the application to be used on the login page.



Figure 22

4.3.11 Use Cases

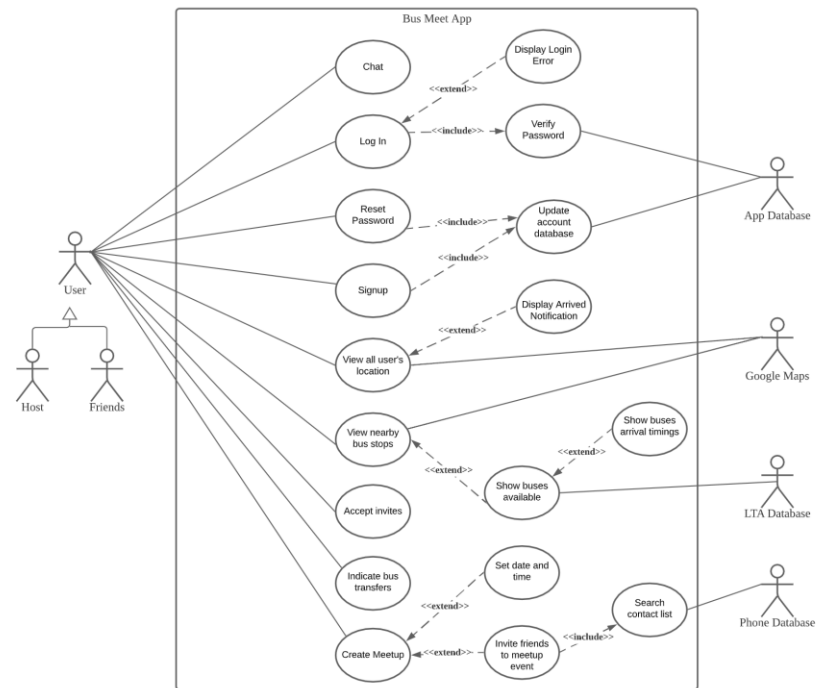


Figure 23

4.4 Challenges

4.4.1 Lack of Experience

Most of us do not have much programming experience in design applications and we had to ensure the application will work on cross-platforms. This means that we cannot use tools like Android Studio that we have used for the previous semester. This situation posed a great challenge to our team as we are all beginners in programming, and we had to learn to use new tools and programming languages.

Since we have chosen React Native to build our project, setting up the environment gave us a lot of trouble. Some of us were not able to set up the coding environment as there are errors either in compiling or starting the app. When we were done with the environment, converting Figma design into codes also created lots of problems for the team. Eventually, the issues were solved as we scheduled meetings to clarify doubts.

4.4.2 Communication

As the team was assigned randomly, naturally communication was not well at the start. At the start of the project, we were all very lost in what we were supposed to do, and the task delegation was messy too. Not everyone knew what they were supposed to do and eventually, everyone started to work on the project individually.

It was after a few meetings with the supervisors, it became clear and thus roles were defined, “Creative team” and “UI/UX”. It took some of us to speak

up and became proactive to take charge and from there we were able to get things going smoothly. Getting back on track to meet our weekly objective/ checkpoints for the project.

4.4.3 Time Restriction

Even though we have 13 weeks to finalize our app, we must learn from scratch to do it. Aside from the time we scheduled for weekly meetings, individuals must self-study by watching hours of tutorial videos on YouTube and manage their workload and commitments for the semester.

It was also challenging to coordinate a suitable timing to schedule a meetup with 11 team members. Thus, most of our meetings were either late at night or on the weekends where we had to sacrifice time with our family

4.4.4 Integration of Features to the main application

As we had to delegate tasks, different components of the apps were done by individual members. Completing the components of the apps was not easy and having the components work together was even more difficult.

We encountered numerous bugs and errors when we were integrating the components into the main app. Some of us have worked hard in coming up with wonderful designs/features for our app. Unfortunately, we were unable to add it due to some constraints. Eventually, we met more frequently, and this integrating process took weeks for us to get settled.

Chapter 5

Conclusion & Recommendation

5.1 Conclusion

Through this project, we have acquired useful technical skills such as learning a new framework like React Native to further strengthen our programming knowledge. We also widened our knowledge on the design aspect of the project such as coming up with a theme for our application to designing wireframes and UI components to implementing them into the application. It was not a smooth journey. However, we learned a lot through our trials and errors and from our peers which in turn strengthened our soft skills that are crucial in collaboration.

For the application, we managed to improve the current two applications which are BusLeh and Glympse by:

- Implementing an integrated function of knowing bus information and chatting with your friends live
- Displaying real-time location
- Improving the overall UX Design of the application
- Introducing splash screens to guide new users

With these new features, we have successfully completed version 2.0 of the application.

However, we feel that we are still unsatisfied with the end product and it could be improved. Therefore, some of us will be coming back after our final exams to work and further improve it to be a complete working app and possibly publish it to the app store.

5.2 Recommendation for future works

No app is perfect and therefore for our project, we feel that there is much more room for improvement. We have many unfinished plans/features to be implemented e.g., animated splash screen, app assistant AI, better UI, and improved backend capabilities.

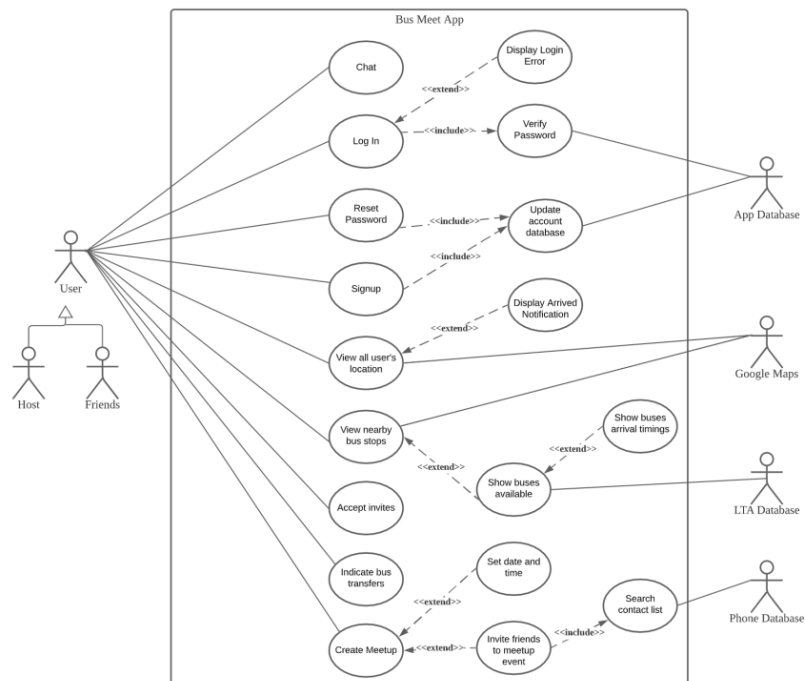
Future works could include the following features:

- Animated splash screen (designed but not implemented)
- Avatar customization (designed but not implemented)
- AI chatbot to assist users on bus information
- More robust real-time information display

Appendices

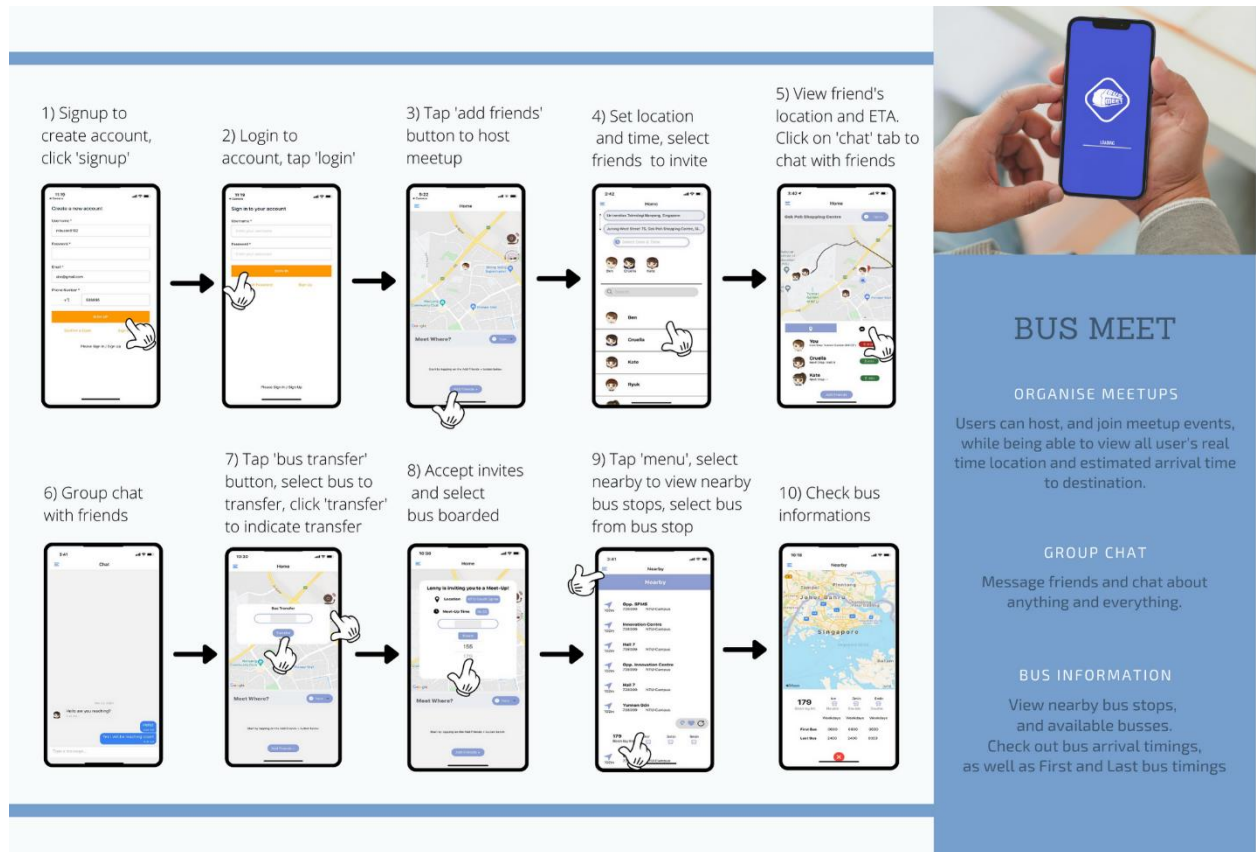
Appendix A: Design Diagrams

- Use-Case Diagram:
 - Link: [https://github.com/lisllin99/IEM-DIP-G4/tree/busmeet_v1/reports/use case diagram/Use Case.pdf](https://github.com/lisllin99/IEM-DIP-G4/tree/busmeet_v1/reports/use%20case%20diagram/Use%20Case.pdf)



Appendix B: User Guide

- Link: [https://github.com/lisllin99/IEM-DIP-G4/tree/busmeet v1/reports/user guide/User Guide.png](https://github.com/lisllin99/IEM-DIP-G4/tree/busmeet%20v1/reports/user%20guide/User%20Guide.png)



Appendix C: Maintenance Guide

- **Fix Bugs:** Fix bugs timely and take care of faults before users discover them. This includes timely checks on whether the services and APIs used on BusMeet are working properly. Furthermore, doing scheduled checks and keeping a lookout for bugs, and fixing them as soon as possible is also the way to make the application more reliable.
- **Security Updates:** Updating security patches to ensure the application is secure and has up-to-date tools to deal with viruses. This will address small issues that may not be important in the given time but will turn into larger problems in the future. By ensuring that the application is secure, it can keep working as desired for as long as possible.
- **Adjusting Features:** In the long run, users may see the need for new features or requirements they would like to see in the application. By constantly adjusting features by including new features and removing features that are not effective will keep the software relevant to the market.
- **Updating Technologies:** These include changes to the operating system, cloud services used, APIs utilized. Software must be updated to meet new requirements so that it runs smoothly.

Appendix D: How-To Guides

- Link: [https://github.com/lisllin99/IEM-DIP-G4/tree/busmeet v1/reports//how to guide/How-To Guide.pdf](https://github.com/lisllin99/IEM-DIP-G4/tree/busmeet%20v1/reports//how%20to%20guide/How-To%20Guide.pdf)

Appendix E: Source Code

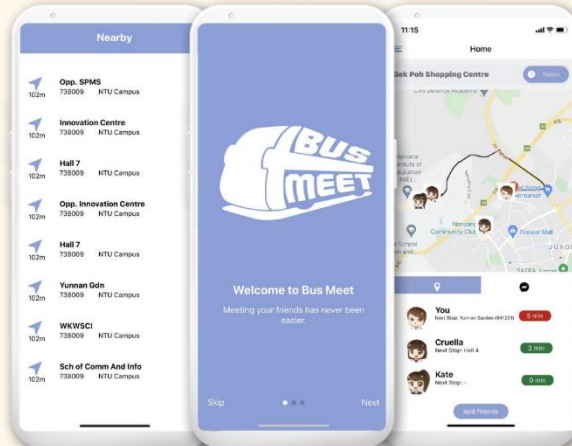
- Link to GitHub repository: [https://github.com/lisllin99/IEM-DIP-G4/tree/busmeet v1](https://github.com/lisllin99/IEM-DIP-G4/tree/busmeet%20v1)

Appendix F: Others

- Poster:
 - Link: [https://github.com/lisllin99/IEM-DIP-G4/tree/busmeet v1/reports/poster/poster.pdf](https://github.com/lisllin99/IEM-DIP-G4/tree/busmeet%20v1/reports/poster/poster.pdf)

BUSMEET

AN ALL-IN-ONE APPLICATION FOR TRANSPORTATION AND COMMUNICATION



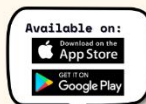
MEET YOUR FRIENDS WITH ZERO HASSLE!

BUS TIMING

Enjoy having your bus arrival timing at your fingertips, completed with information on first and last bus timings.

MEET UP

Invite your friends to meet up with you and track their location and estimated arrival time as you make your way over.



COMMUNICATE

Possibly running late? Use the chat function to inform your friends about your estimated arrival timing!

CUSTOMISE

Stand out with your individuality by customising an avatar that represents you on your screen and your friends'.



Chua Jing Yi, Goh Hongyi, Sebastian, Ho Jia Jun, Huang Bin Melville, Kho Yan Qin, Lee Yi Xuan, Lin Li Swan, Lin Jin Zhao, Loh Jeng, Soo Keng Shuang Serena, Tang Say Sek

- Figma:
 - Link: <https://www.figma.com/file/burrFTjgfZm4VKm99SDVvF/BusLeh-APP?node-id=85%3A2>