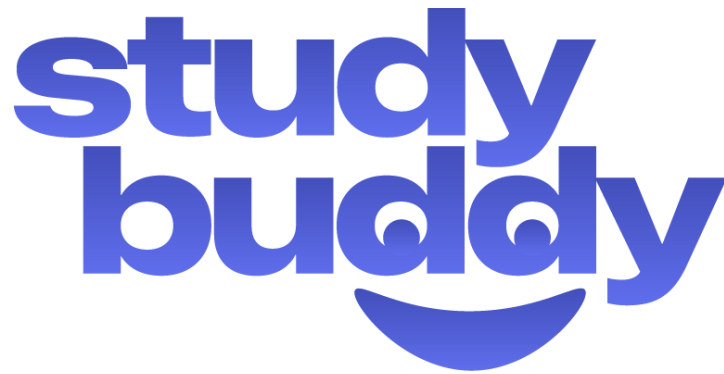


[README]



<https://nusstudybuddy.vercel.app/>

NUS Orbital 2024

Milestone 3

Accidental Study Buddies

Zaidan Sani

Rachel Tai

Contents

Project Summary	4
Proposed Level of Achievement	4
Motivation	4
Aim	4
Scope	5
User Stories	5
Features	6
Timetable	6
Timetable addition	6
Timetable comparison	7
Adding other non-timetable events	7
Booking meetings	8
Timetable summary statistics	9
Slot statistics	9
Course recommender	10
Module planner	10
Live sync to Google Calendar	10
Social Network	11
User authentication	11
User profiles	11
Addition of friends	12
Chat	12
Alternative	12
Mod storage	13
Alternative Plan (Original)	13
Matching algorithms	13
QoL Extensions	14
Light and Dark Mode	14
Email notifications	14
Data export	14
Timetable comparison for non-authenticated users	16
System Design	17
Tech Stack	17
Diagrams	18
Class Diagrams	18
Activity Diagram	19
Use-Case Diagram	19
Wireframes	20
Landing Page	20

Sign in/sign up	20
Timetable Views	21
Social Network	23
Software Engineering Practices	25
Component-Based Architecture	25
N-Tier Architecture	25
Version Control	26
Branching	26
Pull Requests	27
Testing	27
Unit Testing	27
User Testing	28
Integration Testing	28
Regression Testing	28
Bug Fixing	29
Bugs Found and Solutions	29
Development Plan	30
Testing Plan	32
Unit testing of components	32
Unit testing of functions	38
Integration testing	44
Regression testing	46
User testing	50
Methodology	50
Questions	50
Results	51
Beta testing	53
General Problems Encountered	54
Time Constraints	54
Lack of Experience	54
Feature Complexity	54
Limitations	55
Username and email lock-in	55
Username limitation	55
Appendix	56
Features (as from Milestone 1)	56
Timetable Comparison	56
Social Network	56
Quality of Life (QoL) Improvements	57
Links	58

Project Summary

Proposed Level of Achievement

Apollo 11

Motivation

A common challenge that everyone faces in NUS is module planning. The large amount of concerns when it comes to this makes it a more complex issue than expected, due to how much the concerns can vary, and how deceptively difficult it can be to fulfil all the wants that a student needs.

We have seen a lot of attempts to solve some of the problems. A lot of them aim to solve the module planning issue, such as <https://nusplanner.com/> which aims to help students craft their academic plan, and the ever-prominent <https://nusmods.com> which not only lets students see their course information, but also plan out a timetable.

However, a commonality between students is that they would like to take subjects with their friends, or at least find people that they can befriend within their tutorial slots, so that they have some form of support, be it academically or morally, during these sessions. Within our anecdotal experiences, we find that this is a very common problem, and we have had trouble coordinating our timetables with our friends, which can be a slight demotivator for certain students with different comfort zone tolerances.

In addition, when it comes to group work, there is a high chance paired students come from different courses, and may have extremely varied timetables, meaning that there is a difficulty to synchronise timings for meetings.

We believe that collaborative learning is extremely helpful in navigating the complexity of the modules here and believe that solving this issue will be a boon to the students that do utilise the proposed service.

Aim

We first hope to facilitate this coordination by allowing an easy platform for students to share their timetables, and communicate with their friends in order to plan better. This would come in the form of visualisations of overlapping timetables slots, the ability to account for other things such as CCAs, or outside commitments. We also hope to solve some unseen problems, such as distributing the demand away from popular timeslots, by allowing students to see predicted demand and for them to plan for it.

Scope

Summary

StudyBuddy is a module planning platform for students to compare their timetables and find friends to take their courses with.

StudyBuddy is a platform designed to tackle the challenges faced by students at NUS in collaborating module planning with friends. It enhances coordination and promotes collaborative module planning among peers.

StudyBuddy will offer comprehensive timetable comparison features, allowing students to create and manage their timetables, including adding other events. In addition to timetable comparison, StudyBuddy will incorporate social networking features. Students will have user profiles and secure authentication, storing module information for matching algorithms. They can add friends and engage in chats, with a simple matching algorithm extension to help find like-minded peers.

To further enhance the user experience, StudyBuddy will include several quality of life improvements. These will offer a choice between light and dark modes, email notifications, data export options, iCal integration, and the ability to share timetable images.

StudyBuddy facilitates better timetable coordination and communication among students, promoting collaborative learning and addressing the complexities of module planning. By providing visualizations of overlapping slots, incorporating non-academic commitments, and offering predictive insights to distribute demand away from popular timeslots, StudyBuddy seeks to create a supportive and efficient environment for academic planning.

User Stories

1. I want to plan my timetable with my friends.
2. I want to take the same classes with people I am comfortable with.
3. I want to compare my timetable with my friends' timetables easily.
4. I want to see the demand for each time slot, so I can adjust my schedule if needed.
5. I want to add my non-academic commitments to my timetable.
6. I want to find like-minded students who are taking the same modules or tutorial slots as me.
7. I want to communicate comfortably with other students, both online and offline.
8. I want to coordinate meetings and discussions by finding a common free time slot.
9. I want to have a dashboard to view all my lectures, tutorials, meetings, and other academically related activities in one place.

Features

The following section shows the features we have, and the general philosophy behind it. Our original planned feature set can be seen in the [appendix](#).

Note

The terminology on courses and modules are used interchangeably in the documentation here. The programme/degree/major is referred to as aforementioned.

Timetable

Timetable addition

Type

Core

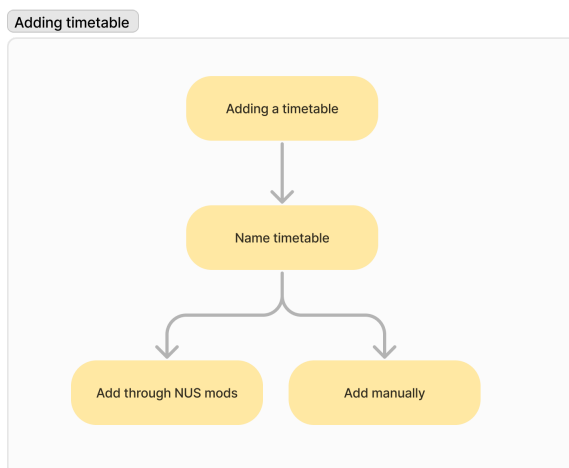
	Projected	Actual
Completion Status	Milestone 1	Milestone 1

This feature refers to the ability to add a timetable to the view.

Philosophy

For convenience, instead of implementing a pure timetable selector, where users would choose their classes for each course, we decided to implement one that would be able to take NUSmods URLs as well. This is as the NUSmods is a commonly used website for all NUS students, as it has been made the main source of course information for all NUS students, and the timetable function is well-refined.

As an extension, we consider implementing a manual timetable creator, given the time allows.



Timetable comparison

Type	Core
------	------

	Projected	Actual
Completion Status	Milestone 3	Milestone 2

Remark: This was originally written as two separate features in the README for Milestone 1 - comparing with someone else (projected Milestone 2) & comparing with multiple other people (projected Milestone 3). Due to the component nature, this was implemented at the same time.

This feature refers to the ability to compare timetables visually.

Philosophy

This feature was the main focus of the project, and is the highlight feature.

The comparison allows for comparison of timetables from three sources:

1. The saved timetables added by the user
2. Saved timetables added by the user's friends
3. A NUSmods timetable

With this implementation, it allows for users to not only compare timetables with their friends who use the website actively, but it also allows users to compare their timetables with their other possible timetables, as well as a NUSmods timetable link, that could be sent by their friends who are not using the application.

The comparison allows for up to four slots, and was limited to such as anything more would be visual clutter. In addition, it would not be effective for the user to compare more than 4 timetables at a singular time.

Adding other non-timetable events

Type	Core
------	------

	Projected	Actual
Completion Status	Milestone 2	Milestone 2

This refers to the ability to add events not directly correlated to the timetables.

Philosophy

This feature was implemented simply as another field in the user data, which allows for easy query utilising Firestore.

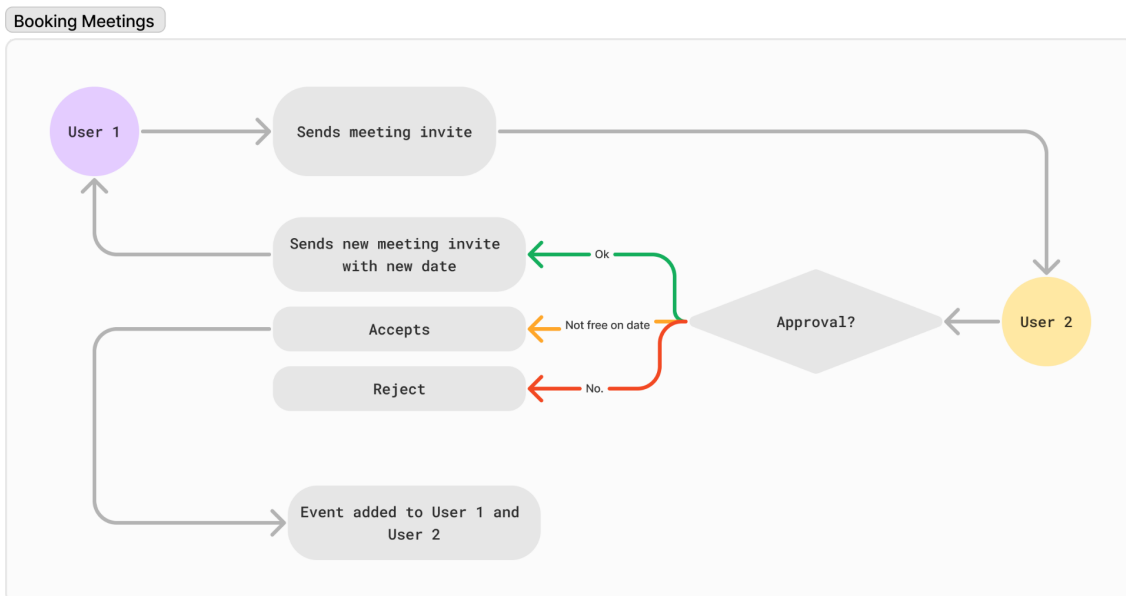
The events are stored separately from the timetables so that they can be accessed easily when the timetable is changed. Since the application allows for storage of multiple timetables, having the events be separate allows for easy change when a different timetable is selected.

Booking meetings

Type	Core						
Completion Status	<table> <tr> <th>Projected</th><th>Actual</th></tr> <tr> <td>Milestone 2</td><td>Moved to future plans</td></tr> <tr> <td colspan="2">shifted to Milestone 3</td></tr> </table>	Projected	Actual	Milestone 2	Moved to future plans	shifted to Milestone 3	
Projected	Actual						
Milestone 2	Moved to future plans						
shifted to Milestone 3							
Reason for Change	The feature is reliant on the social network. Due to time constraints, we were only able to get the social network up pretty late.						

This refers to the ability to add events not directly correlated to the timetables.

Philosophy



This feature is reliant on the Social Network feature.

Using a dropdown, the user can invite multiple of his friends, and then send them a meeting invite. The diagram above shows a simulation of a meeting between two people.

1. When a meeting is created, a new record will be made in the meetings folder. The users involved will then have the meetingID be added into their meeting field, with a status field showing their status (approved, proposed change, rejected)

2. The record in the meetings folder will store these changes in status. Once the entire meeting is approved by everyone, the meeting will be written into their events field.

Timetable summary statistics

Type	Extension
Completion Status	Projected
	Milestone 3
	Moved to future plans
	This was moved to future plans due to a pivot in the project focus to focus on a more polished project.

This feature provides users with summary statistics of their timetable, such as total hours per week, busiest days, and free time slots.

Philosophy

For this feature, the plan is to provide

1. Total hours per weeks
2. Busiest days
3. Free time slots

We plan on utilising user feedback to determine which statistics should be provided.

Slot statistics

Type	Extension
Completion Status	Projected
	Milestone 3
	Moved to future plans
	This was moved to future plans due to a pivot in the project focus to focus on a more polished project.

This feature provides users with slot statistics - where the users are able to see how popular a current slot is for the modules they picked, which may allow them to counter pick.

Philosophy

For this feature, the plan is to provide

1. Popularity of specific class
2. Popularity of timeslot (useful for when there are multiple classes for the same period)

We plan on utilising user feedback to determine which statistics should be provided.

Course recommender

Type	Extension
------	-----------

Completion Status	Projected	Actual
	Milestone 3	Moved to future plans
This was moved to future plans due to a pivot in the project focus to focus on a more polished project.		

This feature provides users with courses (modules) to take based on their previous mods.

Philosophy

We plan on utilising a simple ML implementation for this.

We have yet to confirm the implementation details of this feature.

Module planner

Type	Extension
------	-----------

Completion Status	Projected	Actual
	Milestone 3	Milestone 3

Philosophy

We plan on implementing a very simple module planner - more for visual purposes (no pre-req tracking, no major requirements checking).

Live sync to Google Calendar

Type	Extension
------	-----------

Completion Status	Projected	Actual
	Milestone 3	Scrapped
Scrapped due to resource requirements.		

Social Network

User authentication

Type	Core
------	------

	Projected	Actual
Completion Status	Milestone 1	Milestone 1

Philosophy

As authentication is a very important part, we wanted to make sure we did not do it from scratch as that could result in badly configured security.

We decided to use Firebase Authentication, as it allowed us to integrate it easily with our Firestore database that we used to store user data. The automatic hashing of passwords, allowed us to spend more time implementing the other features, which did not already have a widely accessible implementation in other libraries.

User profiles

Type	Core
------	------

	Projected	Actual
Completion Status	Milestone 2	Milestone 2

Philosophy

We utilised a non-relational database, to allow us to have a more flexible schema with regards to the user details. The potential of the user data stored made us reconsider our original mySQL choice because of the lack of ability to adapt to new requirements if need be.

The user profiles will be used for users to find new friends, and the details provided would be used for the matching algorithm proposed below.

Addition of friends

Type	Core
------	------

	Projected	Actual
Completion Status	Milestone 2	Milestone 2

Philosophy

As the addition of friends is only a two-way thing, we found it acceptable to negate the need of an extra collection, and instead use a value in the user field.

The list of friends is stored with the other user as the key, and the status of the friendship as the value. The value is one of "None" | "Pending" | "Requested" | "Friend", which is used by the application logic to process friend requests and friend approvals.

Chat

Type	Core
------	------

	Projected	Actual
Completion Status	Milestone 2	Milestone 2
The chat is currently working, but we may implement changes to it in the future.		

Philosophy

As a user can have multiple chats, and a chat can have multiple messages, it was not feasible to store the whole chats in the user fields, like done for the friend requests. Hence, only the chatIDs are stored in the user fields.

Thus,

1. If a chat is to be created, it is given a chatID that is then added to the Chats array in both users.
2. The chat object will then contain a chatID, the email / username of both users, and an array of messages.
3. The complete chat object is stored in another collection called "chats", with each document identified by a unique chatID.

Alternative

We considered having the chat be stored twice in both the users data, which would negate the need for a new collection, but it would result in large data duplication, and would not be good for the queries as it would mean there would be 2 writes instead of 1 for each chat.

Mod storage

Type	Extension
Completion Status	Projected
	Milestone 3
	Merged with study plan. Mods are stored through the study plan.
	Actual
	Milestone 3

Philosophy

Storing the mods would allow people to find their friends off the modules they take, allowing it to be used for the matching algorithm.

The modules are now stored under the study plan, instead, which categorises the modules better over the original plan which is mentioned below. This means data is non-duplicated.

Alternative Plan (Original)

The plan is to store the mods in three sections.

1. Past mods
2. Current mods
3. Future planned mods

Past mods will be used for the course recommender while the current and future planned mods will be used for the matching algorithm.

In addition, the mods will also be visible when they are using the basic mod planner, and this will be a small QoL improvement for them to visualise their plan.

Matching algorithms

Type	Extension
Completion Status	Projected
	Milestone 3
	Moved to future plans
	Actual

Philosophy

Currently, we do not have an intricate or complex plan with regards to the algorithms used for the matching. The current plan is to implement simple matching in a checklist manner, and then just returning the results in the sorted order based on how many items they have checked off.

We do eventually plan on implementing simple machine learning through external libraries, as well as using our user surveys to decide what features should be weighted higher.

QoL Extensions

Light and Dark Mode

Type	Extension
------	-----------

Completion Status	Projected	Actual
	Milestone 2	Milestone 2

Philosophy

Due to our standardised approach to using JoyUI, light and dark mode was easy to implement using the theming provided. We just utilised the theming provided and added a button to switch between the modes.

Email notifications

Type	Extension
------	-----------

Completion Status	Projected	Actual
	Milestone 3	Scrapped
Scrapped due to resource requirements.		

Data export

Type	Extension
------	-----------

Completion Status	Projected	Actual
	Milestone 3	Milestone 3

Philosophy

Similar to NUSmods, we plan on implementing export to images (a screenshot of the timetable) as well as export to iCal.

This was added simply using a library which targets a `<div>` element.

Timetable comparison for non-authenticated users

Type	Extension	
Completion Status	Projected	Actual
	Milestone 3	Milestone 3

Philosophy

Anyone can still use the Timetable Comparison function without signing up or logging in. It is available on the landing page.

System Design

Tech Stack



Programming Language
TypeScript

Instead of using just JavaScript, we will use TypeScript, as the type-system and error-checking allows us to create more robust and scalable code that will be vital in maintaining the code as we implement additional features.



Front-End Libraries
React

As mentioned, we will be using React for the front-end, as it allows us to create our desired UI efficiently and promote code reusability, allowing there to be standardisation among all the views.

In addition, the community around React development is large and developed, allowing us to leverage on numerous sources of documentation as well as discussion.



Front-End Libraries
JoyUI

We decided to elect for a front-end component library to help in standardisation across pages and components.

The component library of choice was JoyUI, a variant from MaterialUI, which we chose due to its similarity in appearance to our mockups previously made.



Next.js
React framework



Runtime Environment
NodeJS



Back-End
Firebase Cloud Firestore

We opted for a noSQL database as we wanted to be more flexible with our schema. Originally, we had planned on using MySQL, but after discussion and further research, we realised that the structure of our data would need to not be constrained by a pre-set schema.



Hosting
Vercel

As our app utilises Next.js, it was intuitive for us to elect for Vercel hosting.

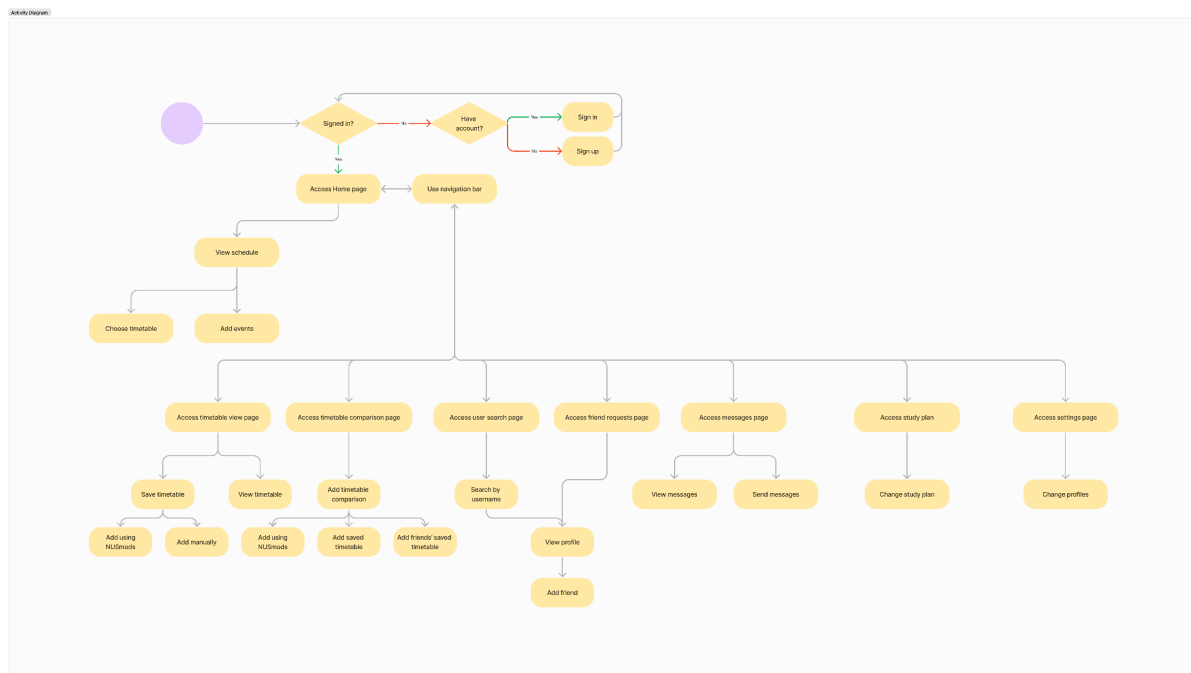
Diagrams

Class Diagrams

Previously, in Milestone 1, we had 2 UML class diagrams. However, we rethought our approach towards the project and ended up focusing towards a functional approach instead, making the class diagram inaccurate, even though it did help us drive the design of the functional programming types.

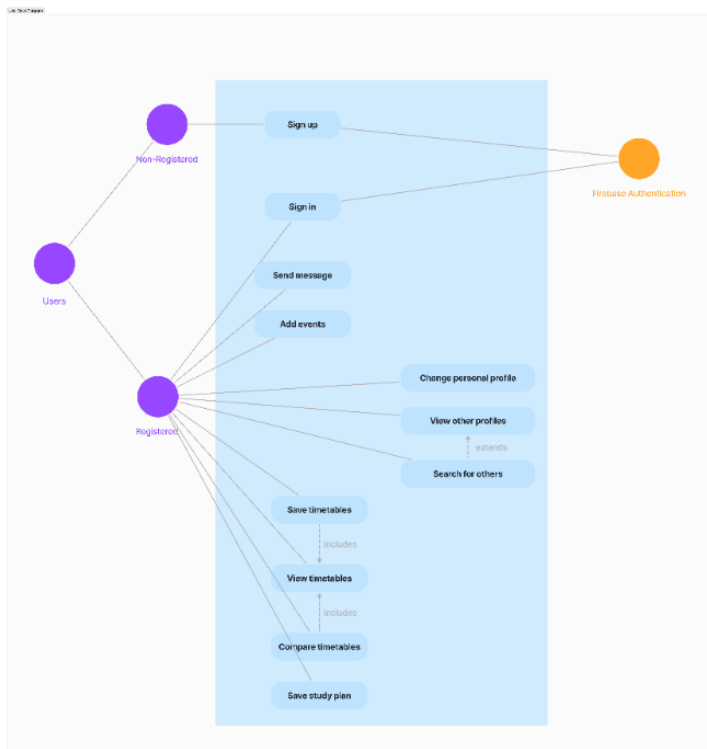
We have removed the diagram as it is now inaccurate to our current system design.

Activity Diagram



This diagram describes the activities the user can undertake.
There are more specific diagrams for each feature listed in the feature portion.

Use-Case Diagram



Wireframes

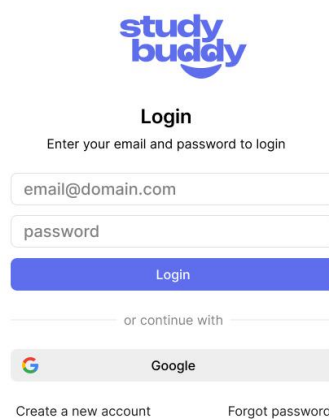
Here are the original design mockups for the UI of the website.

Landing Page



Landing page

Sign in/sign up



[studybuddy]

Log in



Create an account

Enter your email and password to sign up

Sign up

or continue with



Google

Sign in

Timetable Views



timetable displayed

V1 ▼

- Home
- Timetable
- Social
- Messages
- Settings



Add Event

Book A Meeting

Timetable view



timetable displayed
V1

- Home
- Timetable
- Friends
- Messages
- Settings



Add Event Book A Meeting

Event name:

Start Time:

End Time:

Day:...

Book Meeting

Adding an event



welcome to your **dashboard!**

comparing with
@person

timetable displayed
V1

- Home
- Timetable
- Social
- Messages
- Settings



YOUR EVENTS TODAY

MONDAY
Week 13
30/02/2027

1000-1200 **CS1010S**
1600-1800 **MEETING**

with @person

Edit Timetable
Export Timetable

Timetable comparison

Social Network



timetable displayed

V1 ▼

- Home
- Timetable
- Friends
- Messages
- Settings



Add Event

Book A Meeting

Event name:

Start Time:

End Time:

Day: ...

Book Meeting

Chatting



- Home
- Timetable
- Social
- Messages
- Settings

Profile

First name

Jane

Last name

Smitherton

Username

janesmitherton

Gender

Female

Matriculation Year

2023

Email address

email@janesfakedomain.net

Interests

Swimming, Coding

I am looking for a friend taking the course(s)

Profile picture



Change picture

Update Profile

Setting profile



Home

Timetable

Social

Messages

Settings

Connections

FRIEND 1	REMOVE
FRIEND 2	REMOVE
FRIEND 3	REMOVE
FRIEND 4	REMOVE
FRIEND 5	REMOVE
FRIEND 6	REMOVE
FRIEND 7	REMOVE

Friend requests

FRIEND 1	ADD
FRIEND 2	ADD
FRIEND 3	ADD
FRIEND 4	ADD
FRIEND 5	ADD
FRIEND 6	ADD
FRIEND 7	ADD

Find Friends To Connect With

Search

Find New Friends

Gender

Matriculation Year

Interests

Search

Finding friends

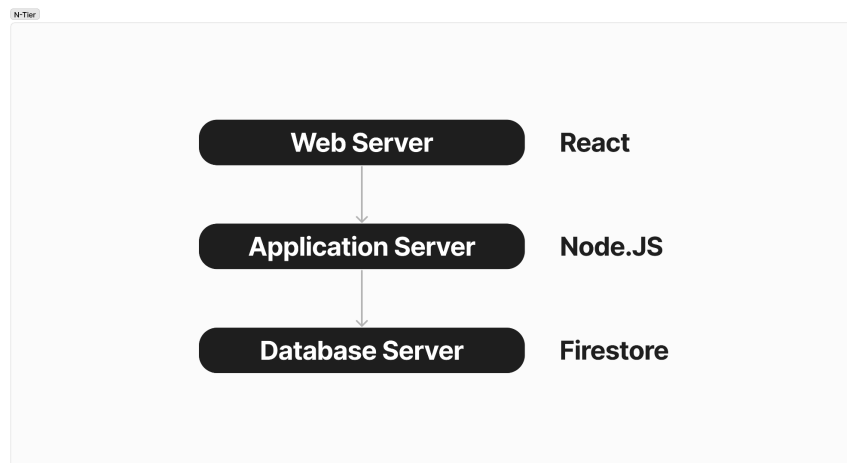
Software Engineering Practices

Component-Based Architecture

We used component-based architecture, where our React web application is built using independent, reusable components. Each component is a self-contained unit that encapsulates its own structure, style, and behaviour. This modular approach makes it easier for us to develop, maintain, and scale our React web application.

N-Tier Architecture

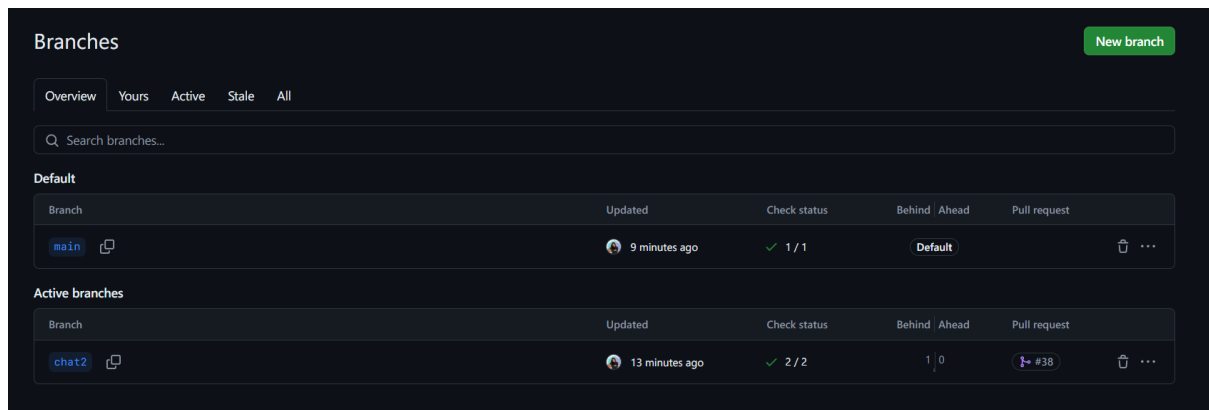
We utilise a three-tier architecture, similar to industry practices with regards to web development.



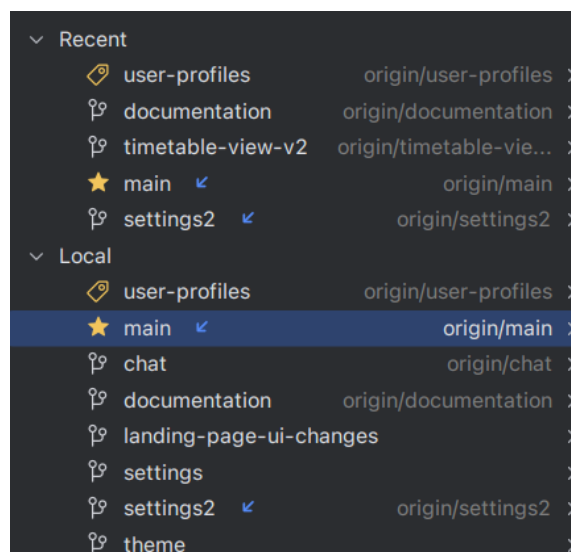
Version Control

Branching

We utilise branches to work on separate features without breaking each other's work unintentionally.

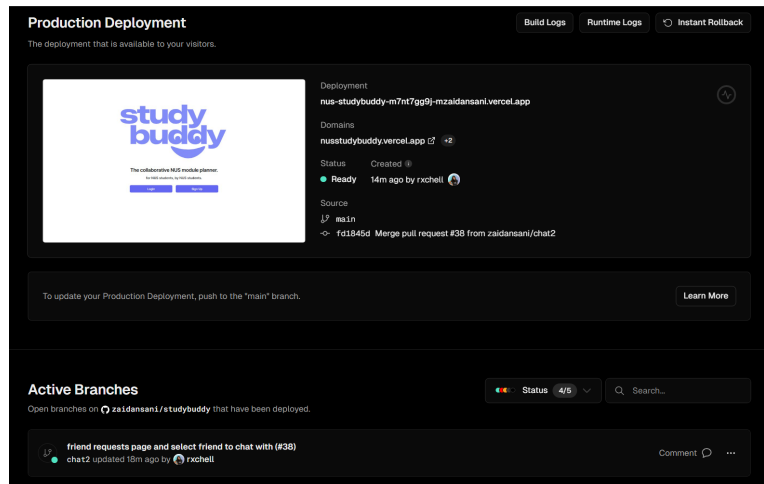


Current branches



History of branches (from WebStorm IDE)

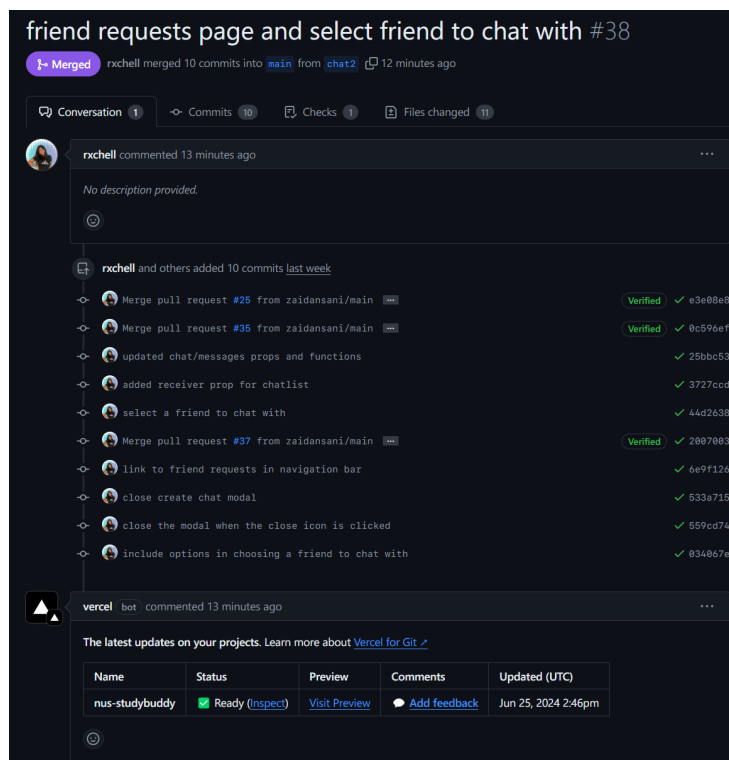
With branches, we are also able to push changes without pushing to the *main* branch, which allows us to review code better. With Vercel, we are able to deploy the branches and see if they build correctly when hosted, which then allows us to double-check that our code runs properly before merging into the *main* branch, using pull requests.



Vercel branches

Pull Requests

We use pull requests to merge the branches. Vercel helps to test if merging causes deployment issues, allowing us to verify that our *main* branch works as intended even after a merger.



Example of pull request to merge branches

Testing

Unit Testing

Jest

We will write and run unit tests for components, functions, and utilities. We will be testing:

- **Rendering:** Ensure components render correctly with different props.
- **State Management:** Test state changes and updates within components.
- **Functionality:** Test various functionalities such as button clicks, form submissions, and user interactions.

User Testing

We will gather feedback directly from users to evaluate the usability and functionality of our product.

- **Usability:** Evaluate how easy the application is to use for its intended audience.
- **Features:** Gather suggestions and insights on what features users find useful or would like to see improved.

Google Forms will be used to create surveys and gather structured feedback from users about their experience with our web application.

Integration Testing

Integration testing verifies interactions between different components or systems to ensure they work together as expected.

We will use **Jest** to test

- **Component Integration:** Test how different components interact and behave together.
- **Deployment Testing:** Ensure the application functions smoothly in its deployed environment.

Regression Testing

We will use **Github actions** for test automation that can be used to automate regression tests.

We will also use **Vercel's Git integration**. The Preview Deployments allows testing of new features and changes before merging to the main branch, with automatic production deployment afterward. There is also Live feedback, with real-time comments on preview environments.

Whenever new features or changes are added to the application, regression tests ensure that existing features continue to work as expected without unintended side effects.

Thus there is **continuous integration and development (CI / CD)** in the development workflow. Whenever there is a pull request or push, the CI / CD pipeline runs the automated tests to ensure our web application can be run and built, and that all functionalities continue working as expected.

Bug Fixing

We were able to spot bugs from multiple sources:

1. Ourselves (while testing and programming)
2. Milestone feedback by other teams
3. User testing forms

Bugs Found and Solutions

Bugs	Solution
User is not redirected to the dashboard upon a successful sign up	• Ensure that the User object is defined and the username property exists
Login, Signup, and Timetable pages do not display error messages	• Added the Alert Status components to show the corresponding error messages • For login and signup pages, the error messages are based on firebase authentication errors
Profile information of receiver are wrongly displayed as the current user in the text bubble and conversation display	• Set the receiver or sender of the text bubble / conversation display based on whether the receiver or sender of the chat is the current user

Development Plan

The development plan will be split into four parts:

1. **Ideation** (pre-milestone 1)
2. **Prototype** (pre-milestone 2)
3. **Extension** (pre-milestone 3)
4. **Refinement** (pre-splashdown)

Ideation				20/05 - 03/06	
Category	From	To	Task	In-charge	
Planning	13/05	25/05	Confirm general system design	Z	R
	13/05	27/05	Confirm UI design	Z	R
	25/05	28/05	Confirm database design	Z	R
Authentication	27/05	02/06	Authentication		R
Timetable	27/05	02/06	Timetable layout	Z	

Prototype				04/06 - 01/07	
Category	From	To	Task	In-charge	
Social Network	10/06	15/06	Chat function		R
	15/06	20/06	Add friend		R
Timetable	03/06	10/06	Add events		R
	03/06	10/06	Collab view for 2 users	Z	
	10/06	17/06	Collab view for >2 users	Z	
QoL functions	20/06	23/06	Light & dark mode		R

Extensions				03/07 - 29/07	
Category	From	To	Task	In-charge	
Social Network	01/07	28/07	Buddy matching	Z	R
QoL functions	01/07	04/07	Email notification		R
	01/07	14/07	Slot statistics	Z	
	04/07	06/07	Data export		R
	06/07	14/07	Basic module planner	Z	
	14/07	28/07	Mod recommenders	Z	

highlighted: The feature was originally planned for milestone 2, but was moved over due to complexity.

Refinement				29/07 - 28/08	
Category	From	To	Task	In-charge	
System testing	29/07	28/08	-	Z	R

Testing Plan

	Tool	Test Details
Unit testing	Jest	Rendering: Ensure components render correctly with different props. State Management: Test state changes and updates within components. Functionality: Test various functionalities such as button clicks, form submissions, and user interactions.
Integration testing	Jest	Component Integration: Test how different components interact and behave together. System Integration: Verify interactions between frontend components and backend APIs. Deployment Testing: Ensure the application functions smoothly in its deployed environment.
Regression testing	Jest and Github Actions	Regression testing is automated and done after each small change. The software is tested again when there is a new push to a branch or a new merge with the main branch.
	Vercel's Git integration	Preview Deployments allow testing of new features and changes before merging to the main branch, with automatic production deployment afterward.
User testing	Google Forms	Gather feedback on real users on the usability and features of our product.

Unit testing of components

Test	Test Details
Landing page	Page renders with the buttons "Login", "Sign Up", and "Compare Timetables" Snapshot testing captures the output of the Page to ensure it renders as expected
Login page	User inputs valid email and password - A green "Sign in Successful!" alert appears - User is redirected to the home page User inputs invalid email - A red "Please use a valid email address." alert appears - User remains in the login page

	User inputs wrong password • A red "Incorrect password." alert appears • User remains in the login page User inputs invalid username • A red "User not found" alert appears • User remains in the login page User does not enter password • A red " Please enter a password." alert appears • User remains in the login page User inputs incorrect email and / or password • A red " Invalid credential." alert appears • User remains in the login page
Sign up page	User inputs valid first name, last name, username, email and password • A green "Sign up Successful!" alert appears • User is redirected to the home page User inputs weak password • A red "Password should have at least 6 characters." alert appears • User remains in the login page User inputs invalid email • A red "Please use a valid email address." alert appears • User remains in the login page User inputs an email which already exists • A red " Account with email already exists." alert appears • User remains in the login page User inputs a username which already exists • A red " Username is already taken." alert appears • User remains in the login page
AlertStatus	• Ensures that <code>AlertStatus</code> renders correctly based on its success prop. It checks for the presence of the correct message and icon. • Uses <code>render</code> to mount the component, <code>screen</code> to query elements, and <code>expect</code> with <code>matchers</code> to assert conditions.
Loading page	User is waiting for the page to load • The page renders the logo and the "Loading..." text
Home page	User is not logged in • User is redirected to the landing page

	User is logged in Page renders the Dashboard component
Navigation Bar	When the user clicks on each item - The user is redirected to the corresponding page The user clicks on Timetable or Social item - Shows the nested pages like View Timetable, Compare Timetables
LightDarkMode	Web application changes from dark / light mode to light / dark mode when this button is clicked
NonAuth Header	When the user is not logged in, the NonAuthHeader component renders the logo and the login and sign-up buttons correctly with the corresponding attributes present
Documentation	Snapshot testing captures the output of the Documentation component to ensure it renders as expected
Timetable Select	User wants to select a timetable and clicks the button Shows a dropdown of all the saved timetables
AddEventDialog	User clicks on the "Add Event" button - Opens a dialog to add a new event User inputs all fields - The event is created Some fields are empty "Please fill in this field" notification appears
ScheduleView	Renders the calendar with the corresponding lessons and events highlighted at the particular date and time
TodayEvents	Renders all the events for the day, otherwise "You have no events today!"
AddTimetableDialog	User clicks on the "Add Timetable" button - Opens a dialog to add a new timetable User enters a name and valid NUSmods url - A green "Timetable added successfully" alert appears User does not enter name and / or description "Please fill in this field" notification appears User does not enter a valid NUSmods url "Failed to add timetable" alert status appears
TimetableView	User wants to see all days of the week

	• Renders a column (Timetable day component) for each day of the week
TimetableDay	User wants to see the time for each day of the week • Renders the text showing the day and the time slots User wants to see when the events are • Time slots are coloured if there is an event during that particular duration
TimetableSlot	User wants to see what type of event • Displays the course code, lesson type, time, and location
Timetable Comparison	User clicks on the "Add Comparison" button • A new column appears for the user to add a new timetable to compare User clicks on the "Remove Comparison" button • The timetable is removed User clicks on the "Save as image" button • The timetable is downloaded as a JPEG
TimetableInput	User wants to add a new timetable using the NUSmods url and clicks the "Change" button • A new timetable is displayed for comparison
TimetableOtherUserSelect	User clicks the "Choose a friend" button • A drop down appears for the user to choose a friend User clicks the "Choose a timetable" button • A drop down appears for the user to choose a timetable to compare
TimetableAccordion	User clicks on "Class List" • A drop down appears for the user to see the list of courses taken by the user's friend
StudyPlan	User clicks on "Save your changes" button • The updated timetable is saved to the user's information
AddModuleSelector	User clicks on the "Add" icon button • The added module will be seen in the study plan
UsernameSearch	User inputs a username and clicks the "Search" button • A list of users with the username would appear in a dropdown
UserDisplay	User clicks on the other user's profile • Redirects the user to the other user's profile page

AddFriendButton	User is not friends with the other user • There is a "Add Friend" button User clicks on the "Add Friend" button • The button changes to "Friend request sent" User has a friend request • There is a "Accept friend request" button User clicks on the "Accept friend request" button • Already added!
AllChats	User inputs a username in the search bar to filter the chats • A list of chats with the corresponding name appears in the chat list User clicks the pencil icon • A new modal pops out for the user to create a new chat with a user User clicks on a particular chat • The corresponding chat appears on the right pane to display the conversation
CreateChat	User selects a friend to chat with • A dropdown of the user's friends appears • The "Start chatting" button appears User already has an existing chat with the friend and clicks on the "Start chatting" button • A red "Chat already exists" alert appears User has no existing chat with the friend and clicks on the "Start chatting" button • A green "Chat created successfully!" alert appears • User is redirected to the messages page
ConversationDisplay	User clicks on the "View profile" button • User is redirected to the other user's profile page
ChatInput	User types a message in the text area and clicks the send icon • The message is sent and a text bubble with the message appears in the conversation
Documentation	User clicks on the search bar to choose a topic • A drop down of a list of topics appears User selects the topic • The information regarding the topic is displayed in the documentation page
Profile	User clicks on the pencil icon



- User can upload a picture to be the profile picture

User clicks on the "Save profile" button and it is successful

- The updated user details are saved
- The saved user details are displayed the next time the user goes to the settings page
- A green "Profile updated successfully!" alert appears

User clicks on the "Save profile" button and it fails

- A red "Profile update failed!" alert appears

Unit testing of functions

Function		Purpose	Test	Expected	Date Passed
utils/Timetable	convertLessonType	Takes in a LessonTypeSmall object and converts it to the corresponding LessonType eg Lecture, Tutorial	convertLessonType(lesson: LessonTypeSmall)	LEC converted to Lecture, TUT converted to Tutorial etc	28/06/2024
utils/Timetable	convertLessonTypeSmall	Takes in a LessonType object and converts it to the corresponding LessonTypeSmall object eg LEC, TUT	convertLessonTypeSmall(lesson: LessonType)	Lecture converted to LEC, Tutorial converted to TUT etc	28/06/2024
utils/Timetable	convertToSemester	Takes in a NUSmodsSemester and converts it to a corresponding number	convertToSemester(semester: NUSmodsSemester)	sem-1 converted to 1, sem-2 converted to 2 etc	28/06/2024
utils/Timetable	convertToCourseSlot	Takes in a CourseParameterisedString and Semester, converting it to a CourseSlot with its parameters lessonType, classNo, and semester	convertToCourseSlot(string: CourseParameterisedString, semester: Semester)	A courseslot is created with lessonType, classNo, and semester	28/06/2024
utils/Timetable	convertToCourseTimetable	Takes in a CourseParameterisedString and Semester, converting to a CourseTimetable type with semester, course, and lessons	convertToCourseTimetable(string: CourseParameterisedString, semester: Semester)	A CourseTimetable is created with Semester, CourseCode, and CourseSlot[]	28/06/2024
utils/Timetable	convertUserCPS	Takes in a user's Course Parameterised String and semester, converting to UserTimetable with the fields semester and timetable	get timetable from CPS	A UserTimetable is created with the fields semester and timetable	23/07/2024
utils/Timetable	getStringfromNUSmods	Ensures that the function correctly extracts the query string from the URL	should return the correct CPS	String format of data is returned from a url	23/07/2024

utils/Timetable	getSemesterfromNUSmods	Returns a NUSModsSemester from the URL	should return the correct semester	A number representing the corresponding semester	23/07/2024
utils/Timetable	getClassesFromTimetable	Converts CourseTimetable and CourseClasses objects to an array of CourseClass objects	get class from timetable	An array of CourseClass objects	23/07/2024
utils/Timetable	getEventsForUser	Converts a UserCPS and Semester into an array of events	get events for user	An array of events sorted by date	23/07/2024
utils/Timetable	getClassesForUser	Converts a UserCPS and Semester into an array of CourseClass	get events for user	An array of CourseClass objects	23/07/2024
utils/Timetable	convertUserCPS	Takes in a user's Course Parameterised String and semester, converting to UserTimetable with the fields semester and timetable	convertUserCPS(cps: UserCPS, sem: Semester)	A UserTimetable is created with the fields semester and timetable	28/06/2024
utils/Timetable	convertSlotToClass	Takes in a CourseSlot and converts it to a CourseClass with parameters like lessonType, classNo, and semester	convertSlotToClass(cs: CourseSlot)	A CourseClass is created with parameters lessonType, classNo, and semester	28/06/2024
utils/parsers	convertToCourse	Transforms a JSON object (jsonData) into a Course object	convert JSON response to course	Returns a Course object containing the code, name, academicYear, and the array of classes	23/07/2024
utils/parsers	convertToCourse	Converts a JSON object representing course data into a Course object.	Convert JSON of a course into events	Returns a Course object containing the code, name, academicYear, and the array of classes	23/07/2024

utils/events	convertCourseClassToEvents	For classes without weeks, assert toHaveLength(0) to ensure the output array is empty	should work on classes without weeks	An empty array, as no events should be generated without week information.	23/07/2024
utils/events	convertCourseClassToEvents	Generates events for a CourseClass with specified weeks	should work on classes with weeks	An array of ClassEvent objects for the specified weeks	23/07/2024
utils/events	convertCourseClassToEvents	To transform course data into a series of events	Convert JSON of a course into events	Returns the array of ClassEvent objects	23/07/2024
services/ChatService	retrieveChats	Takes in a user email and returns all the chats that the user has	retrieveChats(userEmail: string)	Returns an array of chats the user has, in the form of a Chat object	25/07/2024
services/ChatService	retrieveAllReceivers	Takes in a user email and returns the other users whom the user has chats with	retrieveAllReceivers(userEmail: string)	Returns an array of user emails	25/07/2024
services/ChatService	createNewChat	Takes in a sender and receiver to create a new Chat with parameters chatId, sender, receiver, and messages	ChatService	A new Chat is created with parameters chatId, sender, receiver, and messages	09/07/2024
services/ChatService	getChatsLength	Takes in a user email and counts the number of chats the user has	getChatsLength(userEmail: string)	Returns the number of chats the user has	25/07/2024
services/ChatService	chatExists	Takes in a sender and receiver to check if there is a chat between the two users	chatExists(sender: string, receiver: string)	Returns true or false based on whether there is a matching chatId	25/07/2024
services/ChatService	createNewChat	Takes in a sender and receiver to create a new Chat with parameters chatId, sender, receiver, and messages	createNewChat(sender: string, receiver: string)	A new Chat is created with parameters chatId, sender, receiver, and messages	25/07/2024

services/ChatService	updateChatRecords	Takes in a user email and chatId to update the array of chats in the user's document	updateChatRecords(user Email: string, chatId: string)	The chatId is added to the chat field under the User object	25/07/2024
services/ChatService	sendMessage	Takes in a chatId, receiver, sender, and message to be updated in the messages field of the chat	sendMessage(chatId: string, receiver: string, sender: string, message: string)	The message is added to an array of existing messages in the chat	25/07/2024
services/ChatService	markMessageAsRead	Takes in a chatId and receiver to updates the messages in the chat as read by the receiver	markMessageAsRead(chatId: string, receiver: string)	The unread field of the message would be false	25/07/2024
services/ChatService	countUnreadMessages	Takes in an array of Message objects and the receiver's email to count the number of messages which are unread	countUnreadMessages(messages: MessageProps[], receiver: string)	Returns the number of unread messages of a chat	25/07/2024
services/ChatService	getTotalUnreadMessages	Takes in a user email to find the total number of unread messages from all chats	getTotalUnreadMessages(userEmail: string)	Returns the number of unread messages from all the user's chats	25/07/2024
services/ChatService	retrieveChat	Takes in a chatId and returns it as a Chat object	retrieveChat(chatId: string)	Returns all fields of the Chat object	25/07/2024
services/UserService	createUser	Creates a user object given an email, first name, lastName, userName	createUser Function	New User created with the details as given.	09/07/2024
services/UserService	addNewUser	Creates a user object given an email, first name, lastName, userName	addNewUser(user: User)	A new user document is created in firestore, under the "users" collection	25/07/2024
services/UserService	retrieveUser	Get the user's information using the email	retrieveUser(email: string)	Returns a User object	25/07/2024
services/UserService	updateTimetable	Takes in the user's email, NUS mods url, and timetable name to update the array of timetables in the user's document	updateTimetable(email: string, url: string, name: string)	The timetable is added to an array of existing timetables	25/07/2024

services/UserService	updateEvents	Takes in the user's email and an Event object to update the array of events in the user's document	updateEvents(email: string, event: Event)	The event is added to an array of existing events	25/07/2024
services/UserService	retrieveUserTimetables	Takes in a user's email and returns the timetable field	retrieveUserTimetables(email: string)	Returns an array of UserDataTimetable objects	25/07/2024
services/UserService	retrieveUserTimetables	Takes in a user's username and returns the timetable field	retrieveUserTimetablesFromUsername(username: string)	Returns an array of UserDataTimetable objects	25/07/2024
services/UserService	retrieveUserFriends	Takes in a user's email and returns the user's friends	retrieveUserFriends(email: string)	Returns an array of the usernames of the user's friends	25/07/2024
services/UserService	retrieveUserFriends	Takes in a user's email and returns the user's friends	retrieveUserFriends(email: string)	Returns an array of User objects	25/07/2024
services/UserService	retrieveFriendRequests	Takes in a user's email and returns the users with pending friend requests	retrieveFriendRequests(email: string)	Returns an array of User objects with pending friend requests	25/07/2024
services/UserService	retrieveFriendRequestsCount	Takes in a user's email and returns the number of pending friend requests	retrieveFriendRequestsCount(email: string)	Returns the number of friend requests	25/07/2024
services/UserService	retrieveUserEvents	Takes in a user's email and returns the user's events	retrieveUserEvents(email: string)	Returns an array of Event objects	25/07/2024
services/UserService	userHasTimetables	Takes in a user's email and checks if the user has timetables	userHasTimetables(email: string)	Returns true or false based on whether the user has a timetable	25/07/2024
services/UserService	updateUserData	Takes in a User object and updates the fields of the user	updateUserData(user: User)	Updates the fields in the user's document	25/07/2024
services/UserService	retrieveUserFromUsername	Takes in a username and retrieves all the information of the user as a User object	retrieveUserFromUsername(username: string)	Returns a User object	25/07/2024

services/UserService	checkEmailExists	Checks if there is an existing email that is the same	checkEmailExists(email: string)	Returns true if the same email already exists in firestore	25/07/2024
services/UserService	checkUsernameExists	Checks if there is an existing email that is the same	checkUsernameExists(username: string)	Returns true if the same username already exists in firestore	25/07/2024
services/UserService	checkRelationship	Checks the type of relationship between the currentUser and otherUser	checkRelationship(currentUser: string, otherUser: string)	Returns the status of the relationship: None, Friend, Requested or Pending	25/07/2024
services/UserService	sendFriendRequest	Sends a friend request from the current user to the other user if the current relationship is "None"; updates the status for both as "Friend" if current relationship is "Pending"	sendFriendRequest(currentUser: string, otherUser: string)	updates the "friends" fields in each user's document	25/07/2024
services/UserService	acceptFriendRequest	Updates the relationship status between both users as friends	acceptFriendRequest(currentUser: string, otherUser: string)	updates the "friends" fields in each user's document to be "Friend"	25/07/2024
services/CourseService	getCourse	Returns a Course given a Course Code	Get a course	A Course object	23/07/2024
services/CourseService	convertToCourse	Takes in the data from NUSmods and converts it into a course as defined in the type system	convertToCourse(testCourse) JSON data copied from moduleInfo	Course created with the information	23/07/2024

Integration testing

Test	Test Details
Navigation Bar	<code>Navigation Bar</code> component renders the <code>LightDarkMode</code> component Shows the number of friend requests under the Friend Requests item using the function <code>retrieveFriendRequestsCount</code> Shows the total number of unread messages under the Messages item using the function <code>getTotalUnreadMessages</code> Uses the <code>CloseSidebar</code> function to be closed when being toggled in a smaller page width
Dashboard	<code>Dashboard</code> component renders the <code>Navigation Bar</code> , <code>Header</code> , and <code>Schedule</code> components
Header	To ensure that the <code>Header</code> component renders correctly, clicking the menu button should trigger the <code>toggleSidebar</code> function
Schedule	<code>Schedule</code> component renders the <code>TimetableSelect</code> , <code>AddEventDialog</code> , <code>ScheduleView</code> and <code>TodayEvents</code> components
View Timetable page	Page renders the <code>Navigation Bar</code> , <code>Header</code> , and <code>TimetableHome</code> components
Timetable User View	Page renders the <code>AddTimetableDialog</code> , <code>TimetableSelect</code> , <code>TimetableBasicInfo</code> , <code>TimetableView</code> , and <code>TimetableUserClasses</code> components
Compare Timetable page	Page renders the <code>Navigation Bar</code> , <code>Header</code> , <code>NonAuthHeader</code> , <code>TimetableComparison</code> , and <code>TimetableComparisonNoAuth</code> components
Timetable Comparison Cell	Renders the <code>TimetableInput</code>, <code>TimetableView</code>, <code>TimetableAccordion</code>, <code>TimetableSelect</code>, and <code>TimetableOtherUserSelect</code> components Uses the <code>getClassesForUserFromTimetableData</code> and <code>getCoursesForUserFromTimetableData</code> functions to display the user's timetable
TimetableOther UserSelect	Uses the <code>retrieveUserTimetablesFromUsername</code>, <code>getCoursesForUserFromTimetableData</code> and <code>getClassesForUserFromTimetableData</code> functions to show the selected timetable of the user's friend

Study Plan page	Page renders the Navigation Bar , Header , and StudyPlan components
Study Plan	Uses the convertStudyPlanSemesterToString , createDefaultStudyPlan , getUserStudyPlan , and updateUserStudyPlan functions to add and update the user's study plan
AddModuleSelector	Uses the getListOfModules and getCourse functions to add courses to the user's study plan
User Search page	Page renders the Navigation Bar , Header , and Search components
Search	Renders the UsernameSearch and UserDisplay components Uses the getAllUsers function to search for all users for the user to send a friend request
User page	Page renders the Navigation Bar, Header, and AddFriendButton components Uses the checkRelationship and retrieveUserFromUsername functions to show the user's status with the other user, as well as the other user's profile information
Messages page	Page renders the Navigation Bar , AllChats , and Conversation components
AllChats	Uses the retrieveAllReceivers function to search for the list of existing chats the user has Uses the getTotalUnreadMessages function to show the number of unread messages in the chatList
CreateChat	Uses the createNewChat function to create a new chat Uses the retrieveUserFriendsUsers function for the user to select a friend to chat with
Conversation	Renders the TextBubble and ChattingUser components to show the text messages and who the user is chatting with Uses the markMessageAsRead function to update that the message has been read when the user opens the chat Uses the getAllUsersTextBubble component and sendMessage for the user to type a message and send it
Documentation page	Page renders the Navigation Bar , AllChats , and Documentation components

Documentation	Renders the <code>DocumentationSelect</code> and <code>MarkdownLoader</code> components for the user to choose the topic
Settings page	Page renders the <code>Navigation Bar</code> , <code>AllChats</code> , and <code>Profile</code> components
Profile	Renders the <code>ProfilePicture</code> component and <code>retrieveUser</code> , and <code>updateUserData</code> functions to display and update the user's information

Regression testing

Tool	Test Details
Automated testing using Jest and Github actions	- Before a successful push to a branch or merge with the main branch, Github actions will run the Jest tests - The Jest test results (whether the tests pass or fail) are printed in a Common Test Report Format so that the summary results can be shown in Github actions
Vercel's Git Integration	- Whenever there is a push to a branch or merge with the main branch, Vercel will automatically try the run and build the web application - Preview Deployments allows testing of new features and changes before merging to the main branch, with automatic production deployment afterward.

Example of a job that will automatically be run on `Github Actions` whenever there are new changes to the code:

← Workflow file for Jest Report
 ✓ chat service test #33

Re-run all jobs ...

Summary

Jobs

✓ testing

Run details

Usage

Workflow file

testing	
succeeded 1 hour ago in 1m 3s	Q Search logs
> Set up job	1s
> Checkout code	1s
> Setup Node.js	4s
> Install dependencies	40s
> Run tests and generate JSON report	3s
> Print CTRF JSON report	0s
> Run CTRF annotations	3s
> Publish CTRF Test Summary Results	1s
> Publish CTRF Detailed Test Summary Results	2s
> Publish CTRF Failed Test Summary Results	1s
> Publish CTRF Flaky Test Summary Results	1s
> Annotate failed tests	1s
> Post Setup Node.js	0s
> Post Checkout code	0s
> Complete job	0s

Example of failed tests:

Some checks were not successful ×

1 failing and 2 successful checks

✗	 Workflow file for Jest Report / testing (push) Failing after 1m	Details
✓	 Vercel - Deployment has completed	Details
✓	 Vercel Preview Comments -  No unresolved feedback	Details

```
✓ ✗ Run tests and generate JSON report 5s
2143     startTime: '1600',
2144     endTime: '1800',
2145     location: 'COM1-0210'
2146   },
2147   ... 70 more items
2148 ]
2149 }
2150
2151   at Object.log (src/tests/CourseService.test.tsx:5:17)
2152
2153
2154 Test Suites: 1 failed, 11 passed, 12 total
2155 Tests:      18 passed, 18 total
2156 Snapshots:  2 passed, 2 total
2157 Time:       4.731 s
2158 Ran all test suites.
2159 Test results written to: reports/jest-report.json
2160 jest-ctrf-json-reporter: successfully written ctrf json to ctrf/ctrf-report.json
2161 Error: Process completed with exit code 1.
```

Summary of automated test results:

← Workflow file for Jest Report

✔ chat service test #33

Re-run all jobs ⋮

Summary

Jobs

✔ testing

Run details

🕒 Usage

📄 Workflow file

Triggered via push 1 hour ago

👤 rxchell pushed → 5a8ebf0 `jest`

Status

Success

Total duration

1m 12s

Billable time

2m

Artifacts

—

main.yaml

on: push

✔ testing

1m 3s

🔍 − +

testing summary ⋮

Test Summary

Tests 📄	Passed ✔	Failed ✖	Skipped 🛑	Pending ⌚	Other ?	Flaky 🌫	Duration ⌚
13	13	0	0	0	0	0	00:46:54

[A ctrf plugin](#)

Summary

Jobs

✔ testing

Run details

🕒 Usage

📄 Workflow file

Detailed Test Results

Name	Status	ms	Flaky 🌫
get string from NUSmods URL should return the correct CPS	passed ✔	25	
get string from NUSmods URL should return the correct semester	passed ✔	1	
get timetable from CPS should return the correct answer	passed ✔	2	
get class from timetable should return the correct answer	passed ✔	1	
get events for user should return without error	passed ✔	9	
get events for user should return without error	passed ✔	2	
convert JSON response to course should convert as expected	passed ✔	1	
Convert JSON of a course into events should work as intended	passed ✔	7	
ChatService creates a new chat and matches snapshot	passed ✔	3	
Convert a course class into an array of events should work on classes with weeks	passed ✔	1	
Convert a course class into an array of events should work on classes without weeks	passed ✔	1	
renders homepage unchanged	passed ✔	134	
Get a course should work as intended	passed ✔	3	

[A ctrf plugin](#)

Failed Test Summary

No failed tests 🌟

Flaky Test Summary

No flaky tests detected. 🌟

Example of a successful automated deployment:

All checks have passed
2 successful checks

✓  Vercel - Deployment has completed [Details](#)

✓  Vercel Preview Comments -  No unresolved feedback [Details](#)

Deployment | Logs | Functions | Source | Open Graph



The collaborative NUS module planner.
for NUS students, by NUS students.

[Sign Up](#) [Sign Up](#)
[Developer Testimonials](#)

Status
● Ready

Environment
Preview

Duration
2m 12s (2h ago)

Domains

 nus-studybuddy-git-jest-mzaidansani.vercel.app

 nus-studybuddy-6re7uy573-mzaidansani.vercel.app

Source

 jest

 c73e55d firebase setup and signin (to review)

Share

Visit

Deployment Details

Expand All

> Build Logs 2m 2s ✓

> Deployment Summary All 73 1 13 8s ✓

> Assigning Custom Domains Skipped

Example of a failed automated deployment:

Some checks were not successful
1 failing and 1 successful checks

✗  Vercel - Deployment has failed [Details](#)

✓  Vercel Preview Comments -  No unresolved feedback [Details](#)

Deployment Details

Collapse All

Build Logs 1m 30s

All Logs (37) Errors (5) Warnings (0) Find in logs F

Expand 27 Lines

13:46:02.497 Failed to compile.

13:46:02.497 ./src/app/layout.tsx:2:1

13:46:02.498 Syntax error: "next/font" requires SWC although Babel is being used due to a custom babel config being present.

13:46:02.498 Read more: https://nextjs.org/docs/messages/babel-font-loader-conflict

13:46:02.498

13:46:02.504

13:46:02.505 > Build failed because of webpack errors

13:46:02.637 Error: Command "npm run build" exited with 1

13:46:05.317

page 49

User testing

We created and sent a [Google Forms](#) survey to our friends, for them to try out our web application and give feedback.

The survey results are summarised in the "User Testing" tab of  studyBuddy: Testing .

Methodology

We wanted a mixture of quantitative and qualitative feedback - qualitative feedback would be easy to measure, and quantitative feedback would allow us to get more depth with regards to understanding the issues/qualms our users faced.

Questions

User Interface

- The UI was intuitive to use.
- It was easy to find the function I wanted to use.
- The UI was pleasant to the eyes. (design, layout, colour scheme)
- The UI was not cluttered.
- I am satisfied with the interface.
- How quickly were you able to find what you were looking for?
- How easy was it to fill out information and forms on the website?
- How clearly were error messages communicated to you?
- What other features would you like to see for the UI?
- Any feedback on the UI?

Functionality

- Do the functions work as intended?
- If you answered 1 or 2 to the previous question, which functions do not work as intended?
- Do you understand how to use the functions in the website?
- If you answered 1 or 2 to the previous question, which functions do you not understand?
- Which feature did you feel satisfied with? (Multiple select)
- Do you have any recommendations for any of the features?

Timetables

- How easy is it to add a timetable?
- How easy is it to compare timetables?
- How clear and useful are the timetable visualisations?
- How well does the compare timetable feature meet your needs?
- Features to add / Bugs found

Chat

- How easy was it to use the chat feature?

- How intuitive is the chat interface?
- Features to add / Bugs found

Social Network

- How effective is the search function for finding friends?
- How easy is it to add friends?
- How clear are the notifications for friend requests and confirmations?
- Features to add / Bugs found

Documentation

- How clear and understandable is the documentation?
- How easy is it to navigate through the documentation?
- How effective is the search function in finding the information you need?
- How comprehensive is the documentation in covering all necessary topics?
- Features to add / Bugs found

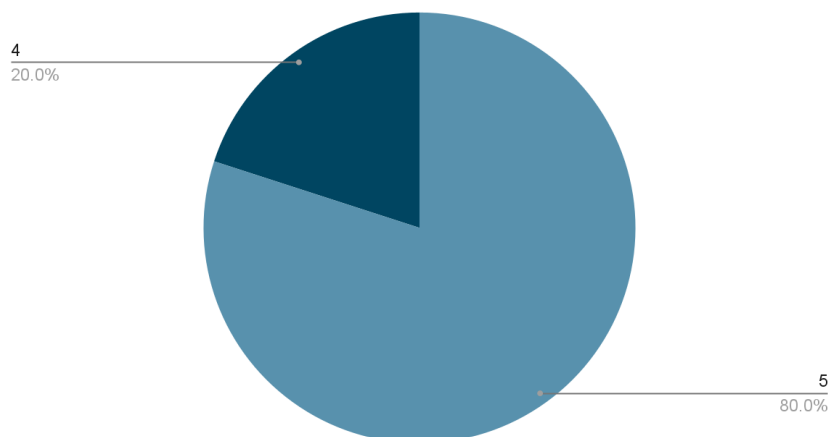
General Feedback

- If you had to describe your experience with the website, what would you say? (3 lines maximum)
- Do you have any feedback you would like to give that did not fit in the sections above?
- How much would you rate your overall experience?
- If the website was polished (and your feedback was taken into consideration and implemented), would you use it?

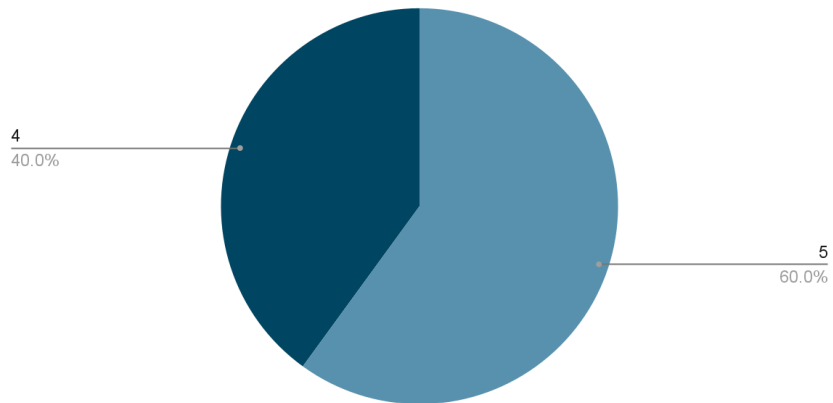
Results

The comprehensive results can be seen in the user testing link, and the following is a curated set of results found most important to us.

How much would you rate your overall experience?



If the website was polished (and your feedback was taken into consideration and implemented), would you use it?



Beta testing

In the context of users planning their modules with NUSMods and using the StudyBuddy website, beta testing involves users from the target audience who will use these tools in their actual study planning environment. This means these users will interact with both NUSMods and StudyBuddy in the way they would normally plan their modules, providing real-world feedback on the usability, functionality, and overall effectiveness of the tools. The goal of this beta testing phase is to identify any issues, gather insights, and make improvements based on how users engage with the tools in their everyday study planning activities.

General Problems Encountered

Time Constraints

As both of us have outside commitments where we are not able to access our laptop/device, as well as collaborate for meetings, we were unable to have meetings as we would have preferred. This also meant that we did most of our work at night.

We circumvented this issue by practicing good software engineering practices - the branching/pull request strategy allowed us to work concurrently on different features.

We also practiced constant communication and checked up on each other, allowing each other to make up for each others' deficiencies.

Lack of Experience

While Rachel has had software engineering practice, Zaidan has not touched full-stack development formally and had to learn a lot of terms specific to the technology he was using, React.

Through the resources provided, constant research, and asking each other, the team was able to learn and figure out their way through problems, such as:

1. Unexpected Firestore quota bursting due to code bug repeating the query constantly

Feature Complexity

The features ended up being more complex to implement than we expected. Thus, in order to have a better and more stable website, we decided to implement less depth into the functionalities and instead focus on having a more complete and robust codebase for the existing features as well as an intuitive UI.

Limitations

Username and email lock-in

Instead of using the UID provided by Firestore, the current linking between the users in Firestore and in Firebase Authentication is by the email, meaning that we have to disable email. This is due to an early mistake where we did not account for the possible change in email, as well as display name. To minimise issues, we did not want to implement a change as it would possibly break all functionality for not much gain.

Username limitation

As we are using Firestore, and the username is used as a key for the friends, it cannot contain dots, as we are using dot notation in our code to push the usernames in. At this point, there seems to be no way to escape the dots, thus we have decided to disallow dots in the username.

Appendix

Features (as from Milestone 1)

Timetable Comparison

Feature	Description
Core features	
Ability to create timetables	Users can manually create their own timetables by selecting courses and scheduling time slots.
Add other events like CCA	Users can add co-curricular activities (CCA) and other non-academic events to their timetable.
Ability to import from NUSmods	Users can import their timetable data from NUSmods for a seamless transition.
Ability to compare with someone else	Users can compare their timetable with another user's timetable to find common free slots.
Extension features	
Compare with more people	Users can compare timetables with multiple users simultaneously to find common free slots for group activities.
Summary statistics	Provide users with summary statistics of their timetable, such as total hours per week, busiest days, and free time slots.
Statistics on slots	Show statistics on how many users have selected particular time slots, helping users understand the demand for specific times.
Course recommender	Suggest courses based on users' interests, schedule gaps, and popularity among peers.
Module planner	Provide a module planning tool to help users plan their modules across different semesters.
Live sync to Google Calendar	Allow users to sync their timetable with Google Calendar for real-time updates and reminders.

Social Network

Feature	Description
Core features	

User profiles	Users can create and customize their profiles, including adding a profile picture, biography, and academic details.
User authentication	Secure login and authentication process to ensure user data privacy and security.
Store mods for matching algorithms	Store information about the modules users are taking to facilitate matching with like-minded students.
Ability to add others into the social network	Users can add friends or connect with other students within the platform.
Chat	There is a messaging function for users to communicate with each other directly through the platform.
Extension features	
Simple matching algorithm	Implement a basic algorithm to match users with similar academic interests, schedules, or module choices for study groups or project teams.
Better matching algorithm	Build on the above matching algorithm to make the friend matching a better experience.

Quality of Life (QoL) Improvements

Feature	Description
Extension features	
Light or dark mode	Users can switch between light and dark mode for a more comfortable user experience.
Email notifications	Send email notifications to users for important updates, deadlines, and reminders.
Data export	Users can export their timetable and other data in various formats for backup or further use.
iCal	Enable users to export their timetable to iCal format for integration with other calendar applications.
Sharing image	There is an option to generate an image of the timetable that can be easily shared on social media or with friends.

Links

Google Forms for User Testing:

<https://forms.gle/t2iQs1d87oncXSKw9>

Testing: studyBuddy: Testing

<https://docs.google.com/spreadsheets/d/14X8v7BSrFVuYmNI5Z9-DICIOUkUMqaSW43wIS-wyvB8/edit?usp=sharing>

Project Log: studybuddy: Project Log

<https://docs.google.com/spreadsheets/d/1CZO1xjuq7CSyLBqIpibbCB8sFt7CoUim0RORQhSBF9w/edit?usp=sharing>

Poster: 6448.jpg

<https://drive.google.com/file/d/14TcOtLrbGxMJvHUljCv6lyU9DxXayw5X/view>

Video: 6448.mp4

<https://drive.google.com/file/d/14Ur7jmWLVuE9TrAB7pmlGzyZdEOWcnY8/view>

