Software Requirements Specification

for

MyCal

Version 1.0 approved

Prepared by Team Xeon

Nanyang Technological University

2 February, 2021

Table of Contents

1. Introduction

1.1 Purpose

MyCal (v1.0) aims to assist the students to plan their courses, share their timetable, find common time slots between different timetables and even explore other courses before planning. It aims to simplify the convoluted process of finding the best-timetable for the user by bringing it to one place. The user can decide what courses (s)he wants to take by browsing through reviews in the discussion forum, and plan a timetable reflecting his/her needs for the semester.

1.2 Document Conventions

The format adapted in the SRS is as follows:

- Main Section Heading:
  - Font: Times
  - Face: Bold
  - Size: 18
- Subsection Heading:
  - Font: Times
  - Face: Bold
  - Size: 14
- Body:
  - Font: Arial
  - Face: Normal
  - Size: 11

1.3 Intended Audience and Reading Suggestions

The document is intended to be used for developers, project managers and testers. The document contains necessary information for the development of the entire system.

For Project Managers and Developers, this document is intended to assist in the maintenance and upgrade of features for the system. It is recommended for developers to begin reading the SRS from Section 1. For Quality Assurance Testers, it is recommended to focus on Section 4 and Section 5, which covers the Functional and Non-Functional Requirements.

1.4 Product Scope

MyCal will benefit students by providing a one-stop solution to timetable planning for NTU Courses. The objective is to take the currently frustrating process and simplify it and make it tolerable. The user will not have to scour through the internet or ask friends for course reviews, but can now find it easily on the website. Additionally, the user can find courses most suitable to his/her style from the review categories, data for which is very hard to find online. The planning functionality will also significantly cut down the time spent trying to manually find appropriate combinations. The flexibility offered in timetable planning will also allow a user to infinitely customize his timetable, accounting for clashes and other obligations.

All of these functions can be done with the click of a button. Gone are the days of manually trying all indexes for hours in STARS. The simplified timetable planning process cuts down on the actual index-allocation part of the planning process, and allows the user to explore more on what courses he/she wants to take, and find courses more relevant to their objectives.

1.5 References

- Express: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction
- React Development: https://reactjs.org/docs/getting-started.html
- MongoDB Development: https://docs.mongodb.com/
- Auth0API: https://auth0.com/docs/api

2. Overall Description

2.1 Product Perspective

MyCal aims to provide users with an easier way to choose their NTU courses and plan their timetable without requiring the effort to go through all the indices of a course while trying to schedule their desired timetable. Currently the NTU stars planner and other websites like NTU VIBE have some functionalities to ease the process of course planning but lack of advanced filtering options such as choosing a free time slot and fixing a particular course index if the user has made up his/her mind to take that particular index are not present which leaves the user spending a lot of time and effort for a simple task to plan their timetable. With MyCal, the user not only gets features such as choosing free time slots, fixing a particular course index, getting all possible combinations of the selected courses but they can also allow clash for particular courses and download their timetable in .ics file to add it to their laptop/tablet or mobile phone calendars.

2.2 Product Functions

MyCal must contain the following functionalities:

1. Users must be able to create an account with their name, email address and password.
2. Users must be able to login using their respective email address and password. The system must prompt the user to register for an account if the customer does not possess an account.
3. Upon successful login, the system must redirect the user to the Home page. The user is then able to navigate to the Plan Timetable, Common free time slot, Discussion forum and home tabs at any time.
4. When the user navigates to the plan timetable tab, the system must display a search bar in the form of a dropdown list where the user can enter the course code/Name and add the valid courses for planning computation. The user must be given the options to choose an index for course, remove the course, add advanced settings and view the computed timetables in weekly format.
5. When the user navigates to the Common Free Time Slot tab, the system must allow the user to upload multiple .ics files and upon clicking the ‘generate’ button, the common

free time slots in all the timetables must be displayed in a weekly format. The user must also be able to view the uploaded .ics files in a weekly format.

6. When the user navigates to the discussion forum tab, the system must display the top rated courses and their description with the option to search for a particular course and filter the courses by the school which offers those courses. The user must be able to view comments, reply/add comments and rate the courses as well.

2.3 User Classes and Characteristics

MyCal's primary user base will be NTU students who want to plan their courses and generate timetables for the semester. Another class of user will be the students who are simply looking for courses to take and can get some insights into the courses from other users comments and ratings.MyCal can also be used for teachers/lecturers/professors to plan their class schedule and download the .ics file to add it into their calendar.

2.4 Operating Environment

MyCal is a web application developed using MERN stack.

2.5 Design and Implementation Constraints

1. The system must be implemented in JavaScript along with HTML and CSS.
2. The system must be developed using MERN stack.
    a. React.js as the frontend
    b. Express.js and Node.js as the backend
    c. MongoDB as the database

2.6 User Documentation

Demo- https://youtu.be/jFZ1npv2xrg

2.7 Assumptions and Dependencies

1. The system uses dx-react-scheduler for displaying timetables.
   Source:https://www.npmjs.com/package/@devexpress/dx-react-scheduler
2. The system uses react-weekly-scheduler to allow users to add their own free time slots.
   Source:https://www.npmjs.com/package/react-weekly-scheduler
3. The system uses nodemailer to send email to the user's email account.
   Source:https://www.npmjs.com/package/nodemailer
4. The system uses bcrypt library to encrypt the user's password for storing.
   Source:https://www.npmjs.com/package/bcrypt
5. The system uses the ics library to create iCalendar events.
   Source:https://www.npmjs.com/package/ics

3. External Interface Requirements

3.1 User Interfaces

The user interface for the website is designed so that it follows the eight golden rules. The website aims for consistency in font, terminology and other design elements. The color scheme is chosen so that the text is readable and the important features stand out. It follows an intuitive flow of actions to be performed by the user.

Figure 1

Figure 1 displays the login and 'create account' screens. The text fields briefly describe the input required so the user is likely to make any errors.

Figure 2

Figure 2 shows the navigation menu. The user can directly go to the desired page from here.

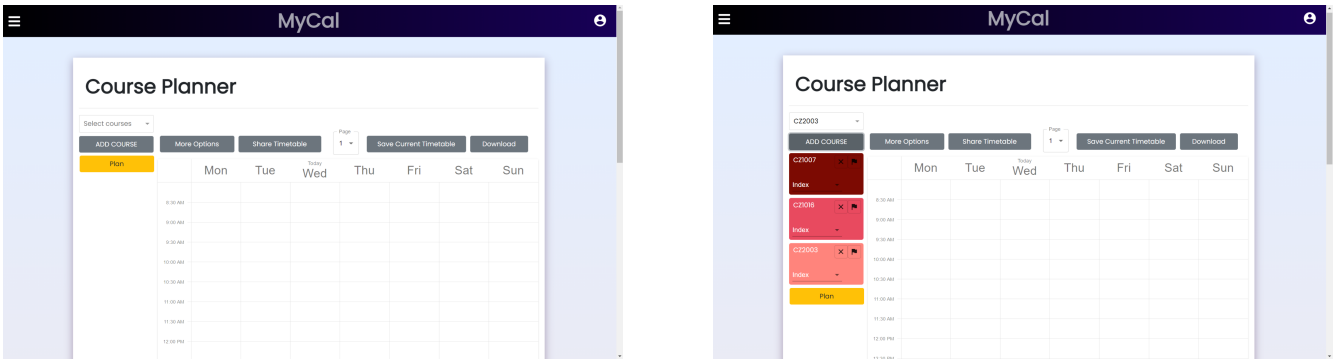


Figure 3

Figure 3 displays the course planning pages. The user first selects the courses to be planned which appear as color coded cards on the side. These cards give the user the option to remove any course or fix a specific index.

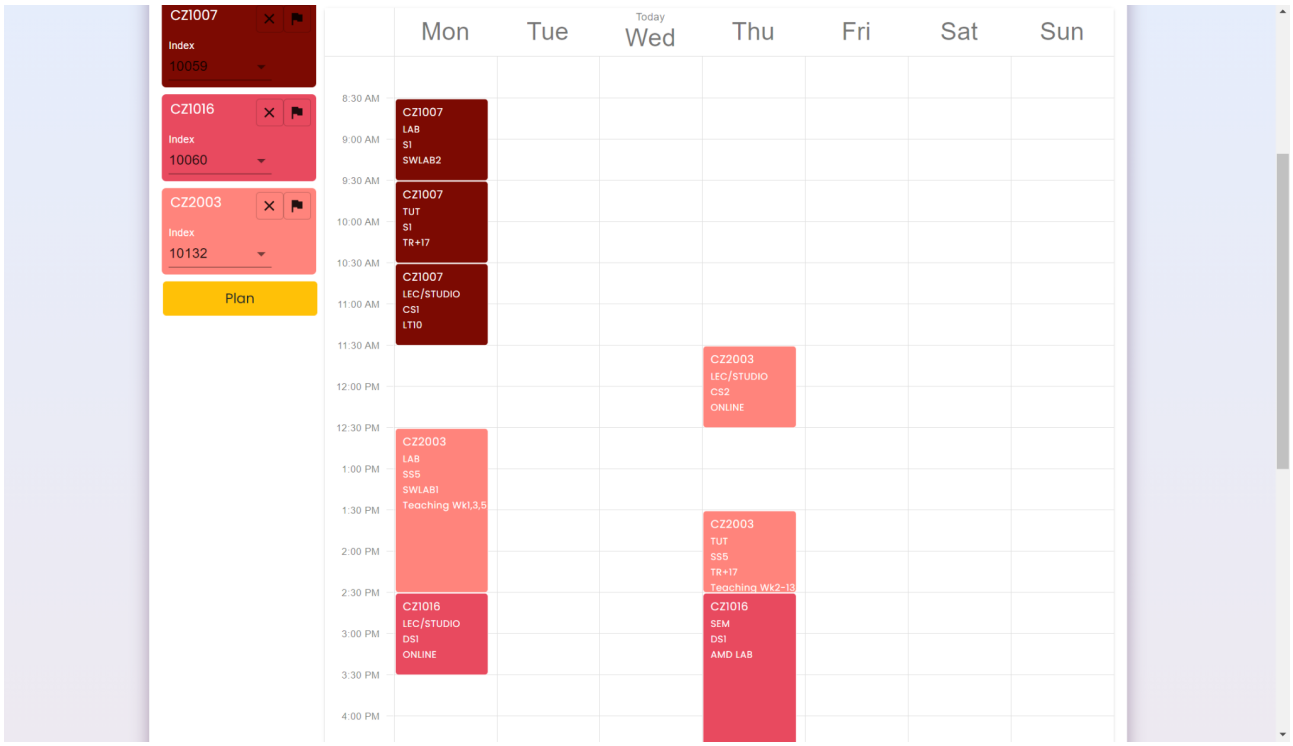


Figure 4

Figure 4 shows how the timetables generated will be displayed.

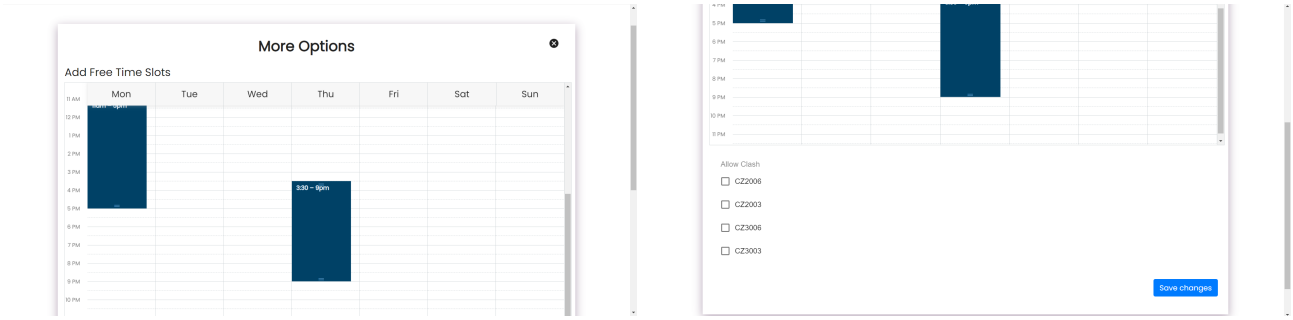


Figure 5

Figure 5 shows the ‘More options’ pop up which allows the user to select time slots which should not be occupied while planning as well as select courses that must allow clashes.

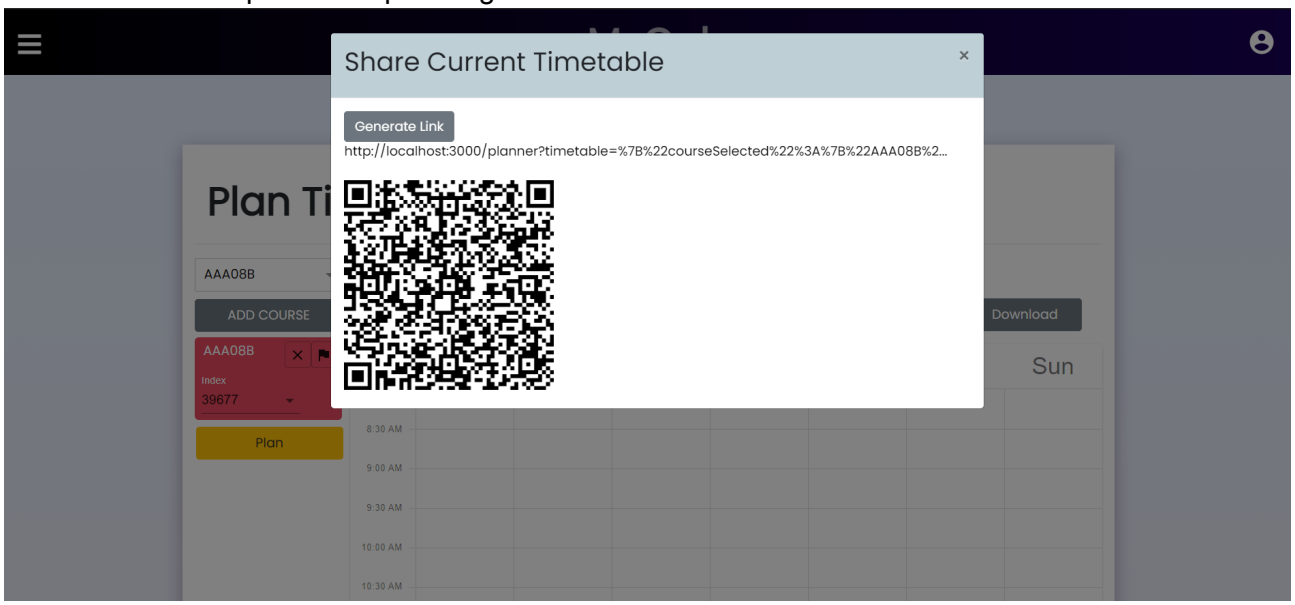


Figure 6

Figure 6 shows the ‘Share Timetable’ pop up which shows the shareable link and a qr code which can be scanned to open the timetable.

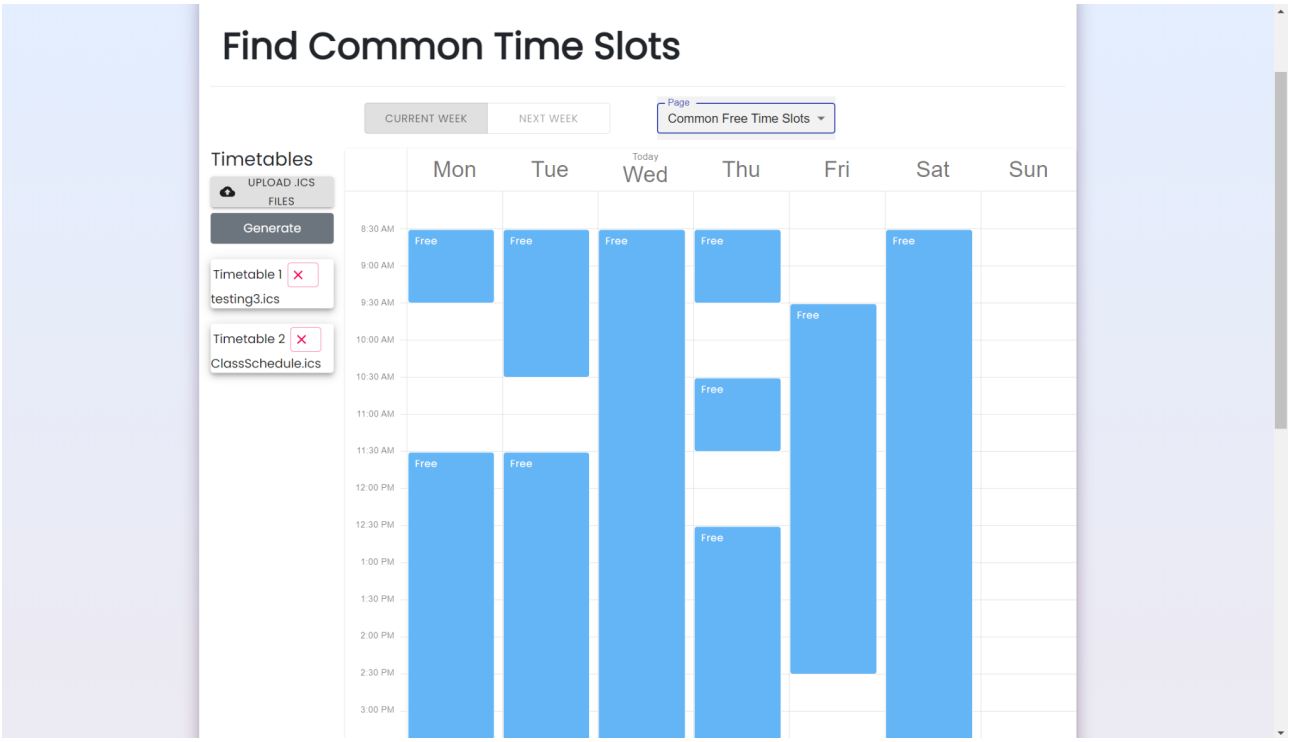


Figure 7

Figure 7 displays the 'Find Common Free Time Slots' page which allows the user to upload ics files by clicking the button. On clicking 'Generate' the common free time slots for current as well as next are shown. The cross can be clicked on the card displayed on the left to remove the timetable.

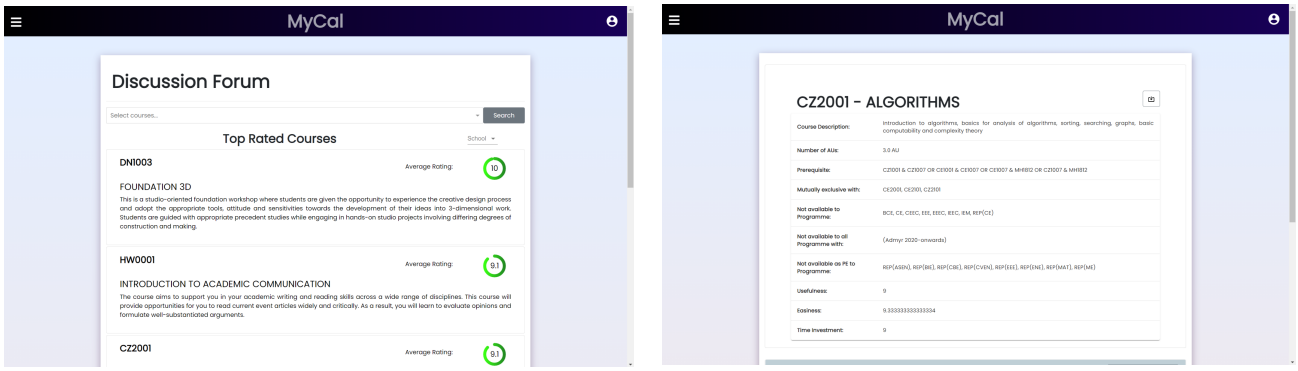


Figure 8

Figure 8 shows the 'Discussion Forum' page, which displays the top 5 rated courses, which can be filtered on the basis of school. On selecting a specific course either from dropdown or the card, the website displays the details of the courses.

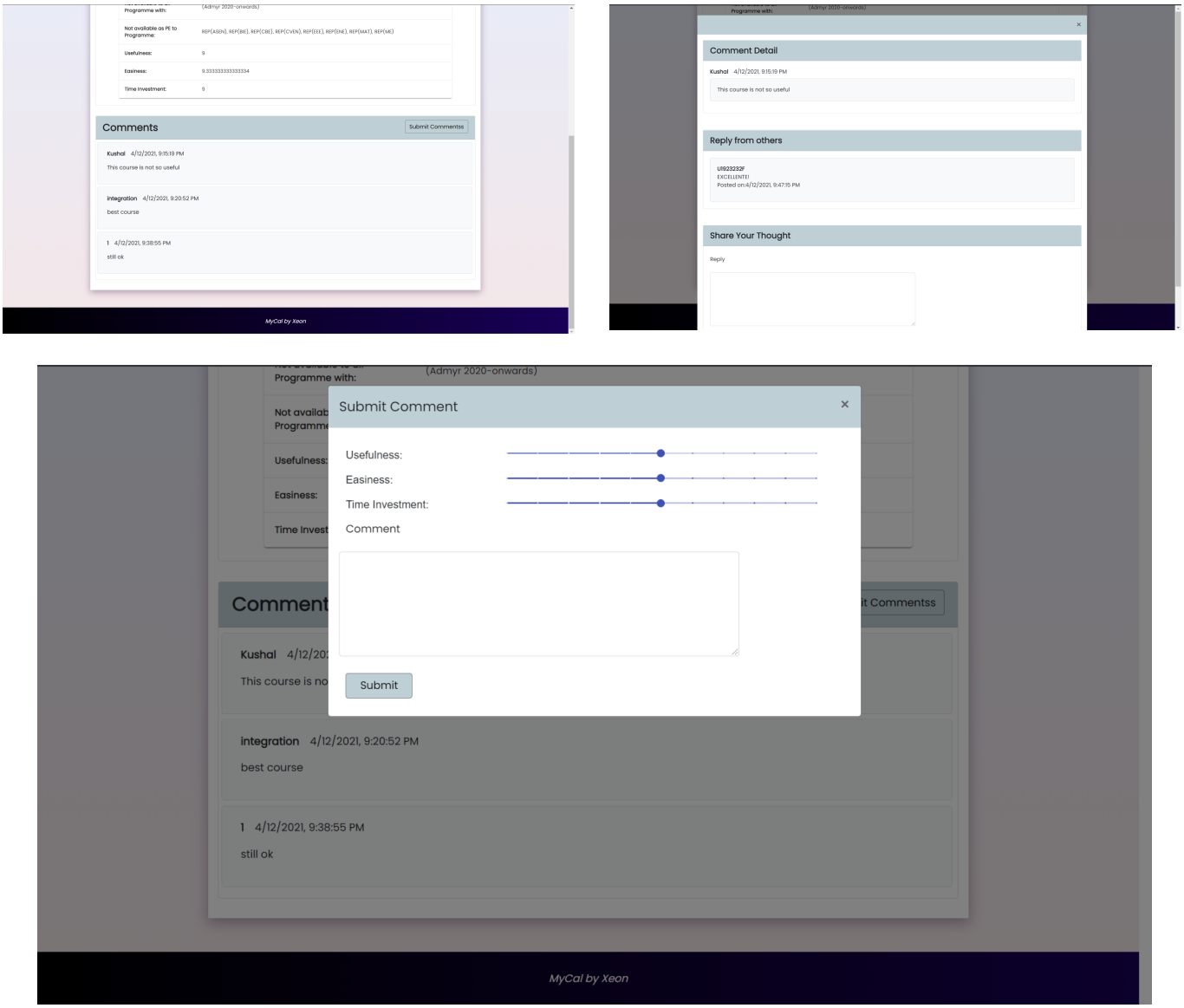


Figure 9

Figure 9 shows how the comments are displayed for every course page. On clicking the comment, the corresponding replies can be seen and the user can add their own reply as well. The user can submit a comment on the course page by rating the course based on the metrics and also add a comment.

Thus, the UI is kept minimalistic and simple so the user can easily navigate through the various features, while highlighting the important features.

3.2 Hardware Interfaces

The website can run on any frequently used browser like Mozilla Firefox, Safari, Google Chrome etc. This includes the hardware which is able to run these browsers. The system must have internet capability. All the information is stored on the cloud database, therefore, it needs to be retrieved while using the website.

3.3 Software Interfaces

The website uses the following software-

1. Windows OS / MacOS - operating system
2. MongoDB Atlas - database
3. Google Chrome, Safari, Mozilla Firefox, Microsoft Edge - supported browsers

3.4 Communications Interfaces

The 'Forgot Password' page uses Nodemailer node module to send the verification code to the user's email. It uses Simple Mail Transfer Protocol (SMTP) communication protocol for mail transmission and focuses heavily on security to minimize chances of interception during the transport.

4. System Features

4.1 Register

The system must allow the user to register.

4.1.1. The system must allow new users to create an account.

4.1.1.1. The user must enter his/her email address.

4.1.1.2. The user must enter the username.

4.1.1.3. The user must enter his/her programme.

4.1.1.4. The user enters his/her year of study in number.

4.1.1.5. The user must enter his/her password.

4.1.1.6. The user must retype to confirm the password.

4.1.1.7. The system must validate the user credentials.

4.1.1.7.1. If the information entered in the email field is in the wrong format, the system must display the error message "User already exists."

4.1.1.7.2. If any field is left blank and the user clicks on Register, the system must display the alert message under the blank field "Please fill out this field".

4.1.1.7.3. If the entered Email has already been registered, the system must display the error message "This email address has already been registered. Please try again."

4.1.1.7.4. If the password contains less than 6 characters, the system must display the error message "The password must contain at least 6 characters. Please try again."

4.1.1.7.5. If the passwords do not match, the system must display the error message "The passwords do not match. Please try again."

4.1.1.8. If the registration is successful, the system must display a 'Your account is registered successfully!' message.

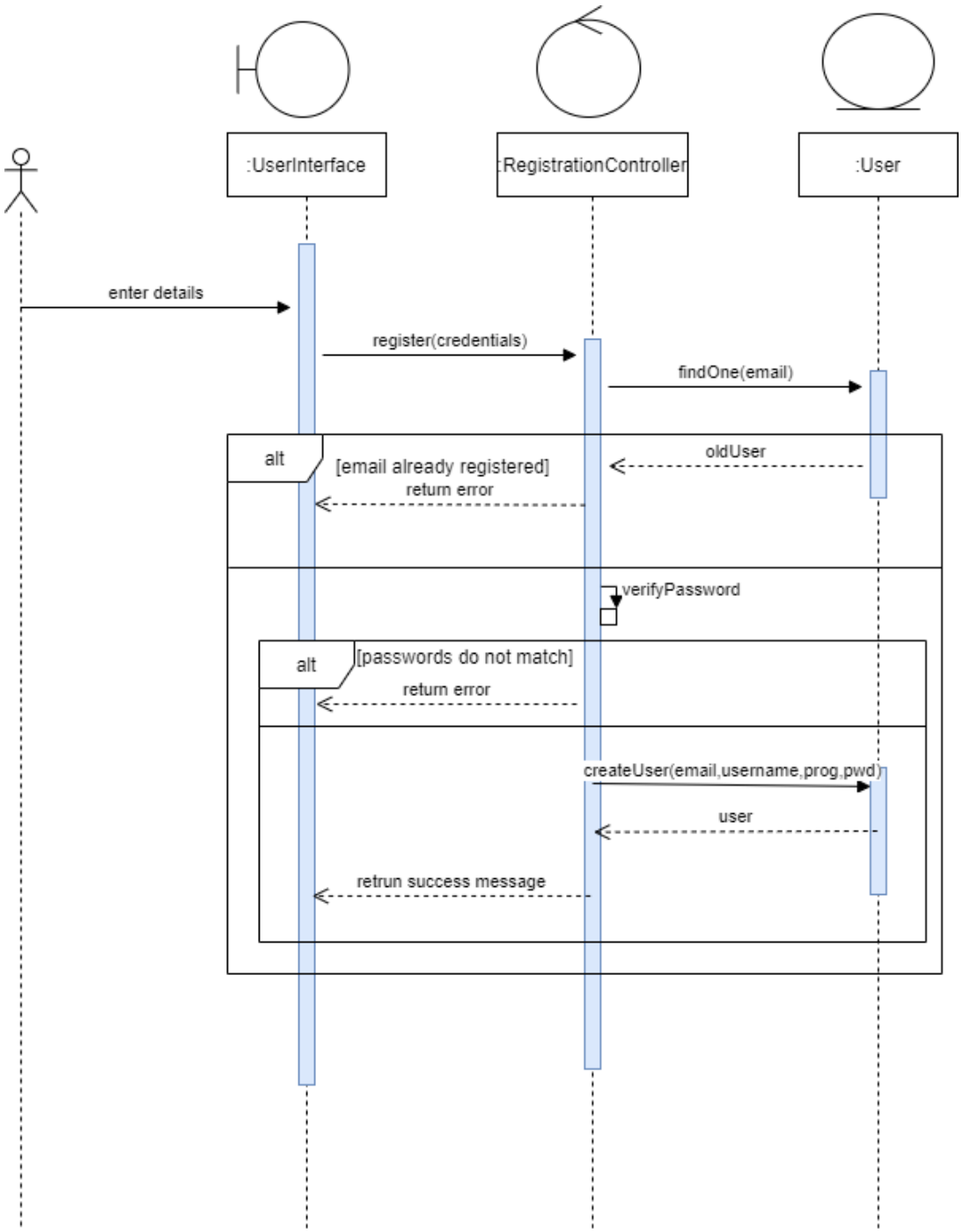
4.1.1.9. The system must redirect the user to the login page upon successful creation of the account.

Use Case Description

Use Case ID:	1		
Use Case Name:	Register		
Created By:	LEE CHIA ZHE	Last Updated By:	Liew Zi Peng
Date Created:	09/02/21	Date Last Updated:	11/04/2021

Actor:	Student
Description:	The user can use this use case to create a new account.
Preconditions:	1. The user is at the 'Create Account' page.
Postconditions:	1. The user successfully created an account or failed to do so.
Priority:	High
Frequency of Use:	Once for every user
Flow of Events:	1. The user enters his/her email address. 2. The user enters his/her username. 3. The user enters his/her programme. 4. The user enters his/her year of study in number. 5. The user enters his/her password. 6. The user confirms his/her password. 7. The user clicks on the "Register" button to submit the details. 8. The system checks that the format of the email address is correct. 9. The system checks that the email address has not been registered before. 10. The system checks that the password's format is correct. 11. The system verifies that the passwords match. 12. The system generates the account in the database. 13. The system displays the message "Your account is registered successfully!". 14. The system redirects to the login page once the account creation is successful.
Alternative Flows:	AF-S9: If the email address is registered before. 1. The system displays the error message "User already exists." 2. Go back to step 1.

	AF-S10: If the password entered contains less than 6 characters, 1. The system displays the error message “The password must contain at least 6 characters. Please try again.” 2. Go back to step 6. AF-S11: If the passwords do not match. 1. The system displays the error message “The passwords do not match. Please try again.” 2. Go back to step 8.
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	The user has an email address.
Notes and Issues:	



Sequence Diagram for Register

4.2 Login

4.2.1. The user must be able to log-in.

4.2.1.1. The user must enter his or her email address.

4.2.1.2. The user must enter his or her password.

4.2.1.3. The system must validate the user credentials.

1.3.1. If the email address is not in the database, the system must display the error message “User doesn't exist.”

1.3.2. If the password is wrong, the system must display the error message “Invalid credentials.”

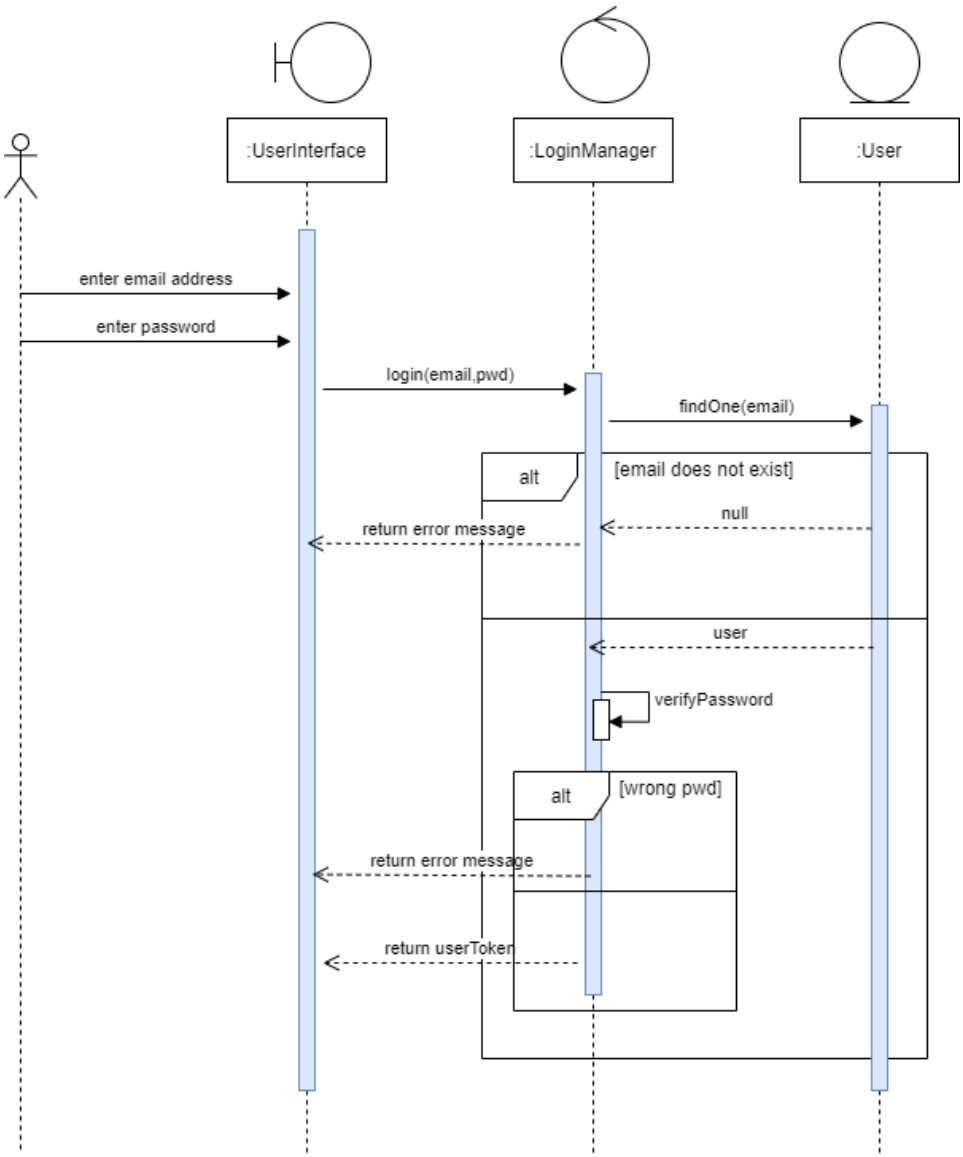
4.2.1.4. The system must redirect the user to the home page upon successful log-in.

Use Case Description

Use Case ID:	2		
Use Case Name:	Login		
Created By:	Liew Zi Peng	Last Updated By:	Liew Zi Peng
Date Created:	09/02/21	Date Last Updated:	11/04/2021

Actor:	Student
Description:	The user can use this use case to login to their account
Preconditions:	1. The user has created an account. 2. The user is navigating the login page.
Postconditions:	1. The user logs in successfully or fails to log in.
Priority:	High
Frequency of Use:	Once per session
Flow of Events:	1. The user enters the user’s email address. 2. The user enters the user’s password. 3. The user clicks on the ‘Login’ button to submit credentials. 4. The system verifies if the email address exists in the database. 5. The system verifies if the password is correct. 6. The system redirects the user to the home page upon successful login.
Alternative Flows:	AF-S4: If the email address does not exist in the database, 1. The system displays the error message “User doesn't exist.” 2. Go back to step 1. AF-S5: If the password is incorrect, 1. The system displays the error message “Invalid credentials.”.

	2. Go back to step 2.
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	



Sequence Diagram for Login

4.3 Forgot Password

4.3.1. The user must be able to reset his/her password using the 'Forgot Password' function.

4.3.1.1. The user must enter his/her Email address.

4.3.1.1.1. If the email address is not in the database, the system must display the error message "User does not exist."

4.3.1.2. The system must send a verification code to the email account and display the message "A verification code has been sent. Please check your email."

4.3.1.3. The user must enter the code into the system.

4.3.1.3.1. If the code entered is wrong, the system shall display the error message "Wrong code. Please try again."

4.3.1.3.1.1. If the user enters the verification code wrongly for 3 times, the system must display the error message "Limits exceeded to enter the verification code.

4.3.1.4. The user must enter the new password and the confirming password.

4.3.1.4.1. If the password contains less than 6 characters, the system shall display the error message "The password must contain at least 6 characters. Please try again."

4.3.1.4.2. If the passwords do not match, the system shall display the error message "The passwords do not match. Please try again."

4.3.1.5. The system must display the message "Password updated."

4.3.1.6. After the user confirms, the system must redirect the user to the home page.

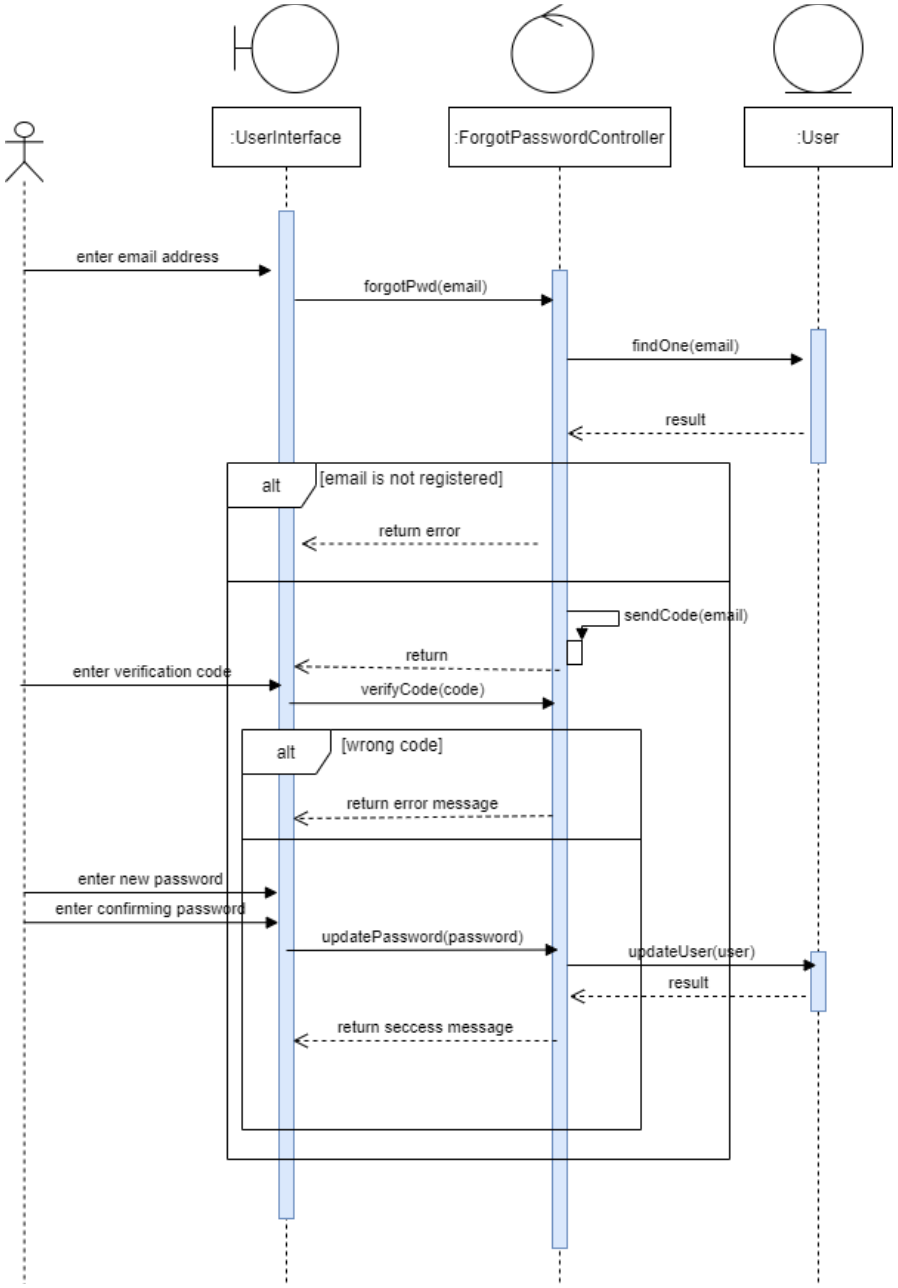
Use Case Description

Use Case ID:	3		
Use Case Name:	Change Password (Forgot Password)		
Created By:	Liew Zi Peng	Last Updated By:	Liew Zi Peng
Date Created:	09/02/21	Date Last Updated:	11/04/2021

Actor:	Student
Description:	The user can use this use case to reset the user's password when they have forgotten his or her password.
Preconditions:	1. The user has created an account. 2. The user is navigating the login page.

	3. The user clicked on “Forgot Password?” and the system redirected the user to the forgot password page.
Postconditions:	1. The user resets the password successfully or fails to do so..
Priority:	High
Frequency of Use:	Seldom
Flow of Events:	1. The user enters the user’s email address. 2. The system checks the format of the email address. 3. The system verifies that the email address exists in the database. 4. The system generates a 6-alphanumeric-characters verification code. 5. The system sends the verification code to the user’s email account. 6. The system displays the message “A verification code has been sent. Please check your email.” 7. The user enters the verification code. 8. The system verifies the code entered. 9. The user enters the new password. 10. The system checks the password’s format. 11. The user enters the confirming password. 12. The system verifies that the passwords match. 13. The system updates the user’s latest password in the database. 14. The system displays the message “Password updated”. 15. The system redirects the user to the login page.
Alternative Flows:	AF-S3: If the email address does not exist in the database, 1. The system displays the error message “User does not exist.” 2. Go back to step 1. AF-S8: If the verification code is wrong, 1. The system displays the error message “Wrong code. Please try again.” 2. Go back to step 6. AF-S10: If the password entered contains less than 6 characters, 1. The system displays the error message “The password must contain at least 6 characters. Please try again.” 2. Go back to step 8. AF-S12: If the passwords do not match with one another, 1. The system displays the error message “The passwords do not match. Please try again.”. 2. Go back to step 10.
Exceptions:	EX-S8: If the user has entered the verification code wrongly for 3 times,

	1. The system displays the error message “Limits exceeded to enter the verification code.”. 2. Go back to step 1.
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	



Sequence Diagram for Forgot Password

4.4 Plan Timetable

4.4.1. The system must allow the user to plan their semester time table.

4.4.1.1. The user must be able to select the courses for planning.

4.4.1.1.1. The system must allow the user to select the courses in the following manner.

4.4.1.1.1.1. The user must enter the course code in a search bar.

4.4.1.1.1.1.1. If the course code does not exist, the system must display the error message “The course does not exist. Please try again.”

4.4.1.1.1.2. The list of selected courses is displayed and updated.

4.4.1.1.2. The system must allow the user to remove any of the selected courses.

4.4.1.2. By default, the system must display all possible combinations of the indices that prevent clashes among courses.

4.4.1.2.1. The system must display an error message “Exam clash between [courseCode] and [courseCode]” if the user has added courses with conflicting exam schedules i.e same exam date and time.

4.4.1.2.2. The system must display the error message “No such timetable possible.Cannot allot course because of clash [courseCode]” if the user has added a course which clashes with another course.

4.4.1.2.3. For each result generated, the timetable from Monday to Friday and the list of course codes with their indexes must be displayed.

4.4.1.2.4. The user must be able to switch from one result to another using a dropdown list

4.4.1.3. The system must allow the following filtering options for the timetables displayed.

4.4.1.3.1. The user must be allowed to enter the time slots where he or she does not want to have any class.

4.4.1.3.2. The user must be allowed to choose the index he or she wants to fix for any course from a drop-down list.

4.4.1.3.2.1. The fixed indexes flag icon must be highlighted with a different colour.

4.4.1.3.3. The system must be by default in the “Minimum Clash” setting.

4.4.1.3.3.1. The user must be able to select courses for which clash is allowed.

4.4.1.3.3.2. The system must allow at most 4 courses clashing with each other.

4.4.1.3.3.3. The timetables are displayed in ascending order from least conflicts to most conflicts

4.4.1.3.4. The system must display the error message “No such timetable possible.Cannot allot course because of clash [courseCode]” if no such timetable is possible.

4.4.1.4. The system must allow the user to save timetables to his/her account.

4.4.1.5. The user must be able to view the saved timetables and modify them.

4.4.1.6. The user must be able to remove the saved timetables.

4.4.1.6.1. The system must get the user’s confirmation before removing the saved timetables.

4.4.1.7. The user must be able to export and download a timetable as a file in “.ics” format.

Use Case Descriptions

Use Case ID:	4		
Use Case Name:	Plan Timetable		
Created By:	Astha	Last Updated By:	Liew Zi Peng
Date Created:	09/02/21	Date Last Updated:	13/4/2021

Actor:	Student
Description:	The student plans the timetable.
Preconditions:	1. The user is at the “Plan Timetable” page.
Postconditions:	1. The user planned the timetable.
Priority:	High
Frequency of Use:	More than once a day during course registration and add drop period
Flow of Events:	1. The user enters the course code/course name in the search bar. 2. The system displays intermittent matching results in a drop down list. 3. The user selects the course and clicks the “Add Course” button” 4. The system will display the list of selected courses. 5. The user clicks the ‘Plan’ button. 6. The system generates all pages with each possible timetable combination with corresponding indexes that prevent time clash. Each timetable is displayed in a weekly calendar format.
Alternative Flows:	AF-S1: If the user enters incorrect course code/name, 1. The system will not show any result in the drop down list. 2. Go back to step 1. AF-S4: If the user removes any course from the ones selected, 1. The system will remove the selected course. 2. Go back to step 4.
Exceptions:	EX-S6: If no timetable can be generated,. 1. The system displays the correct error message related to the reason why no timetable combination can be generated. 2. Go back to step 4.

Includes:	Use Case 5, 6, 7 and 10
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	5		
Use Case Name:	Set Filtering Options		
Created By:	Astha	Last Updated By:	Liew Zi Peng
Date Created:	09/02/21	Date Last Updated:	13/4/2021

Actor:	Student
Description:	The user filters/customizes the timetable according to their needs
Preconditions:	1. The user is at the “Plan Timetable” page.
Postconditions:	1. The user filtered the timetable according to the needs.
Priority:	High
Frequency of Use:	Frequently
Flow of Events:	1. The user selects specific indexes for some of the courses which are to be fixed by clicking on the flag button of the course on the “Plan Timetable page”. 2. The user clicks on the ‘More Options’ button on the “Plan Timetable” page. 3. The system displays a “More Options” dialog modal. 4. The user selects the time slots (or all slots, i.e. entire day) that he/she wants to keep free on that week day under the “Add free time slots” option. 5. The user selects the course he/she wants to allow time clash for under the “Allow Clash” option.

	6. The user clicks on the “Save Changes” button to save the changes. 7. The system updates the advanced options for planning timetable.
Alternative Flows:	AF-S1-7: At any step, if the user clicks on the cancel button of the dialog modal, 1. If changes are made, the system will display the confirmation dialog on whether the user wants to discard the changes. If the user wants to discard the changes, the “More Options” dialog modal will be closed. 2. If no changes are made, the “More Options” dialog modal will be closed.
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	6		
Use Case Name:	Save Timetable		
Created By:	Liew Zi Peng	Last Updated By:	Liew Zi Peng
Date Created:	13/4/2021	Date Last Updated:	13/4/2021

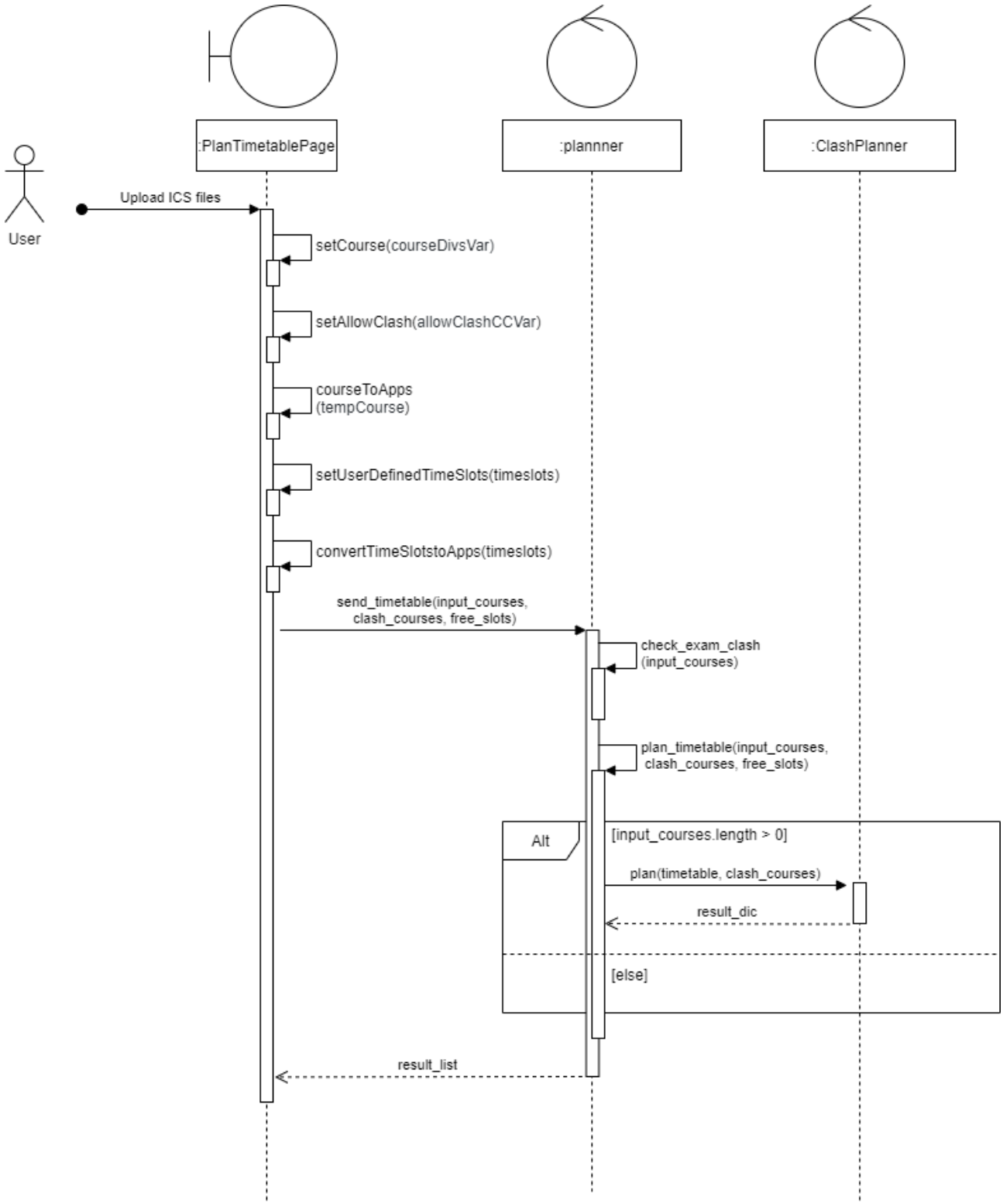
Actor:	Student
Description:	The user can use this use case to save the timetable
Preconditions:	1. The user is at the “Plan Timetable” page. 2. The user planned the timetable. 3. The user has logged in.
Postconditions:	1. The selected timetable is saved into the user’s account.
Priority:	Medium

Frequency of Use:	Average
Flow of Events:	1. The user clicks on the ‘Save Current Timetable’ button. 2. The system saves the current timetable into the user’s account. 3. The system displays the message “Current timetable has been saved.”
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	7		
Use Case Name:	Download Timetable ICS file		
Created By:	Liew Zi Peng	Last Updated By:	Liew Zi Peng
Date Created:	13/4/2021	Date Last Updated:	13/4/2021

Actor:	Student
Description:	The user can use this use case to download the timetable in ics file format.
Preconditions:	1. The user is at the “Plan Timetable” page. 2. The user planned the timetable.
Postconditions:	1. The selected timetable is downloaded in ics file format
Priority:	Medium
Frequency of Use:	Average

Flow of Events:	1. The user clicks on the ‘Download ICS file’ button. 2. The system generates the ics file based on the current timetable. 3. The ics file is automatically downloaded to the user’s default download folder.
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	



Sequence Diagram for Plan timetable

4.5 Find Common Free Time Slots

4.5.1. The system must be able to find the common free time among at most 10 timetables.

4.5.1.1. The user must be able to import multiple timetable files in ".ics" format.

4.5.1.2. The system must display the timetables uploaded in weekly calendar format.

4.5.1.3. The system must display the common free time slots in a new timetable.

Use Case Description

Use Case ID:	8		
Use Case Name:	Upload ics files		
Created By:	Liew Zi Peng	Last Updated By:	Liew Zi Peng
Date Created:	13/4/2021	Date Last Updated:	13/4/2021

Actor:	Student
Description:	The user can use this use case to upload ics files to view the ics files in weekly calendar format.
Preconditions:	1. The user is at the 'Find Common Free Time Slots' page.
Postconditions:	1. Every uploaded ics file is allocated one page and is displayed in weekly calendar format.
Priority:	Medium
Frequency of Use:	Average
Flow of Events:	1. The user clicks on the "UPLOAD .ICS FILES" button. 2. The user can choose multiple ICS files to upload. 3. The system displays the list of uploaded ICS files with the name of each ICS file.
Alternative Flows:	AF-S3: If the user remove any uploaded ICS file from the list, 1. The system will remove the selected ICS file from the list 2. Go back to step 3.
Exceptions:	
Includes:	

Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	9		
Use Case Name:	Find Common Free time		
Created By:	Akshat Sharma	Last Updated By:	Liew Zi Peng
Date Created:	09/02/21	Date Last Updated:	13/4/2021

Actor:	Student
Description:	The user can use this use case to import multiple timetables to search for common free time slots
Preconditions:	1. The user is at the ‘Find Common Free Time Slots’ page. 2. The user has used the use case 8 to upload the ics files.
Postconditions:	1. Users uploaded timetables are saved and processed by the system.
Priority:	Medium
Frequency of Use:	Average
Flow of Events:	1. The user clicks on the ‘Generate’ button. 2. The system processes the imported timetables and generates the common free time slots and stores it on the ‘Common free time slots’ page in the dropdown menu. 3. The user navigates to the ‘Common free time slots’ page using the page dropdown menu to view the common free time slots.
Alternative Flows:	
Exceptions:	EX-S2: If there are no common free time slots among the imported timetables. 1. The system displays the message “There are no common free time slots among the uploaded ics files. Please try a different set of schedules.”

Includes:	Use case 8
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.6 Share Timetable

4.6.1. The system must allow the user to share their timetable.

4.6.1.1. The user must select the timetable they want to share from a dropdown menu.

4.6.1.2. The system must give the following options for sharing.

4.6.1.2.1. The system must generate a sharable link which can be copied directly.

4.6.1.2.2. The system must display a qr code for scanning.

4.6.1.3. Upon opening the link, the system must direct the user to the “Plan Timetable” page and display the shared timetable correctly in a weekly calendar format.

Use Case ID:	10		
Use Case Name:	Share Timetable		
Created By:	Akshat Sharma	Last Updated By:	Liew Zi Peng
Date Created:	09/02/21	Date Last Updated:	13/4/2021

Actor:	Student
Description:	The user can use this use case to share the timetable generated.
Preconditions:	1. The user is at the “Plan Timetable” page.
Postconditions:	1. The selected timetable’s link and QR code are generated.
Priority:	Medium
Frequency of Use:	Average
Flow of Events:	1. The user clicks on the ‘Share Current Timetable’ button. 2. The system displays a dialog modal. 3. The user clicks on the “Generate Link” button on the modal.

	4. The system generates the link which encodes the information of the current timetable. 5. The system displays the link and the corresponding QR code.
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.7 Discussion Forum

4.7.1. The system must allow the use of a discussion forum.

4.7.1.1. The user must be able to search for a course by course code or course name in a search bar.

4.7.1.1.1. If no course is related to the user input, the system must display the error message “Please select a valid course before adding.”

4.7.1.1.2. The system must display the valid course in a new page along with the course name, course description, course rating, course prerequisites, number of academic units, mutually exclusive courses, grading type, not available to programs, usefulness, easiness and time investment.

4.7.1.1.3. The system must display the possible search results in a drop-down list below the search bar.

4.7.1.1.4. The system must display the top rated courses by default if no course is chosen.

4.7.1.1.5. The system must display the course code, average course rating and course description.

4.7.1.2. The system must display the course name, course code, number of academic units, school, description, pre-requisites, mutually exclusive courses, grading type, time of final exam and user posts in the course page.

4.7.1.3. The user must be able to save the course for future reference.

4.7.1.3.1. The user must be able to view the saved courses from their profile button.

4.7.1.3.2. The system must allow the user to “unsave” the saved course.

4.7.1.4. The user must be able to create a post for a specific course.

4.7.1.4.1. The system must offer the user an option to rate the usefulness of the course.
4.7.1.4.2. The system must offer the user an option to rate the difficulty of the assessments in the course.
4.7.1.4.3. The system must offer the user an option to rate the time investment in the course.
4.7.1.4.4. The user must be able to write a comment which does not exceed 512 characters.
4.7.1.4.4.1. The system must prevent use of any profanity.
4.7.1.5. The user must be able to reply to another post.
4.7.1.5.1. The comment must not exceed 512 characters.
4.7.1.6. The system must display the course of study which the user takes besides the username in the post or comment.

Use case description

Use Case ID:	11		
Use Case Name:	Recommend modules i.e. gerpe/UE		
Created By:	Kushal	Last Updated By:	Lee Chia Zhe
Date Created:	09/02/21	Date Last Updated:	13/4/2021

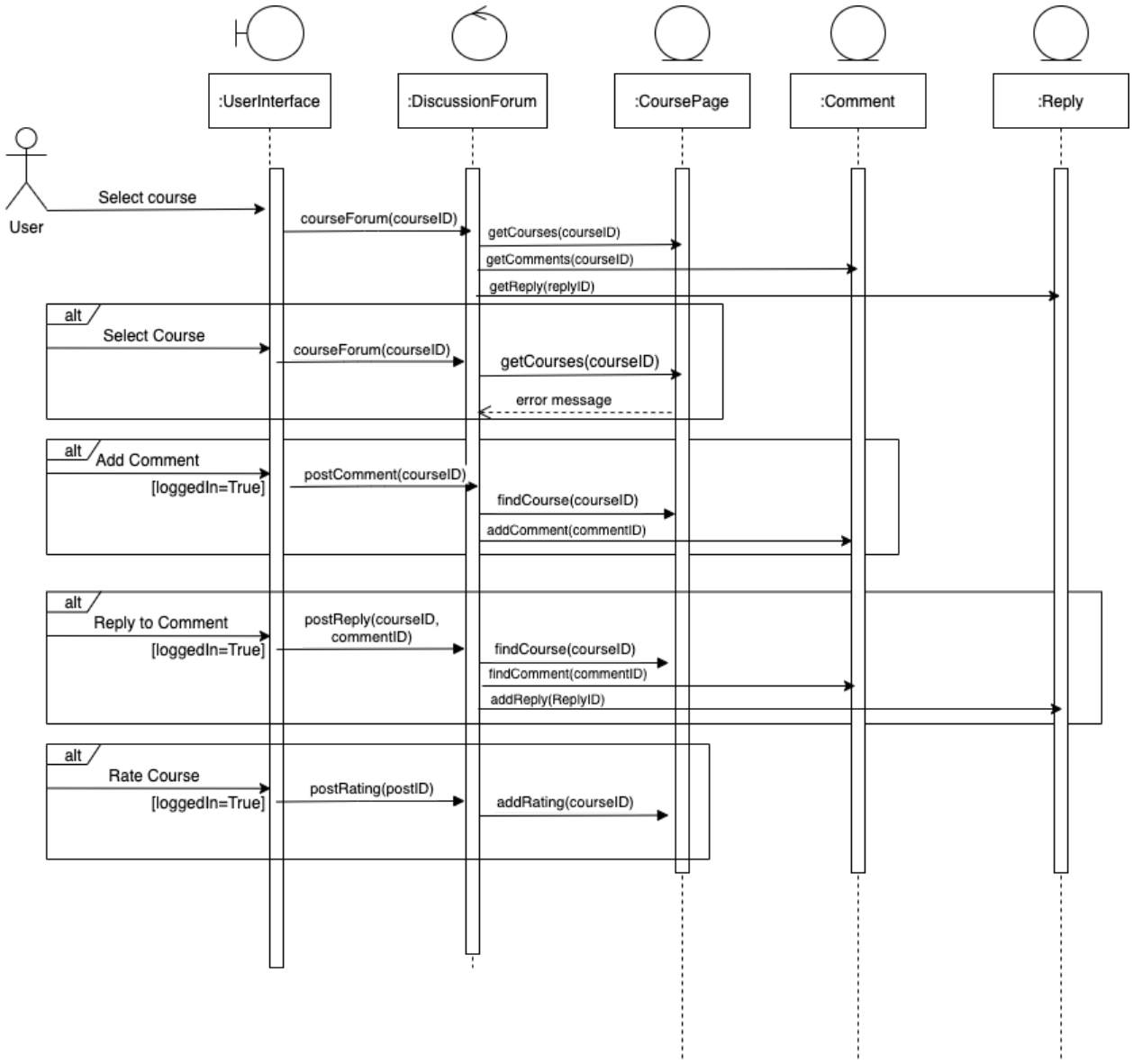
Actor:	Student
Description:	The system recommends modules to the users.
Preconditions:	1. The user is at the Discussion Forum page.
Postconditions:	1. The user acquired the information about courses and comments.
Priority:	High
Frequency of Use:	Frequently
Flow of Events:	1. The Discussion Forum page displays the following options for the user: a. A search bar to visit the course description and comments. b. Top rated courses by users, sortable by school and type of course (i.e. UE/ GERPE).

	2. The user clicks the 'More Options' button on the homepage and chooses the school and type of course to get filtered results. 3. Clicking on any of the course link leads to the discussion forum page for the course
Alternative Flows:	
Exceptions:	EX-S1: 1. The system must display a blank page with only the search bar when no courses have any ratings
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	12		
Use Case Name:	Viewing course pages and create comments		
Created By:	Kushal	Last Updated By:	Lee Chia Zhe
Date Created:	09/02/21	Date Last Updated:	13/4/2021

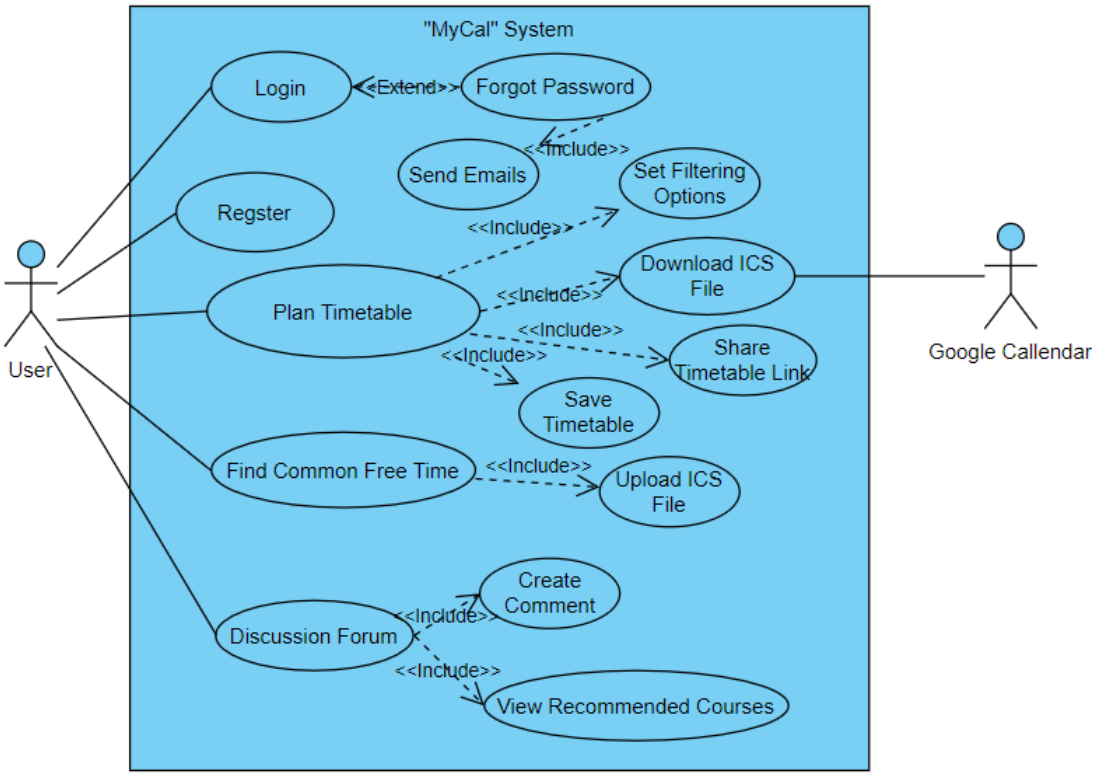
Actor:	Student
Description:	The user views and comments on courses.
Preconditions:	1. The user has logged in. 2. The user is at one of the course pages.
Postconditions:	1. Comments created by the user are stored into the database. 2. Replies to comment created by the user are stored into the database

Priority:	High
Frequency of Use:	Frequently
Flow of Events:	1. The Discussion Forum page displays the following options for the user: a. Search bar to visit the course description and discussion page b. Top 5 rated courses by all other users. 2. The user searches for the course code/course name in the search bar. 3. The System displays the page for the course which contains the: a. The course information b. The course reviews c. The comments left by other users about the course 4. The user clicks the “Submit Comment” button to create a new comment in the thread. 5. The user clicks on the comment left by other users to view the comment detail. 6. The user replies to the comment reply by other users by typing in the space below the “Share Your Thought”.
Alternative Flows:	AF-S4: The user saves the course by clicking the save button at the top right of the course detail page. 1. The system must save the course to the saved list of saved courses by the user 2. Go back to step 3.
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

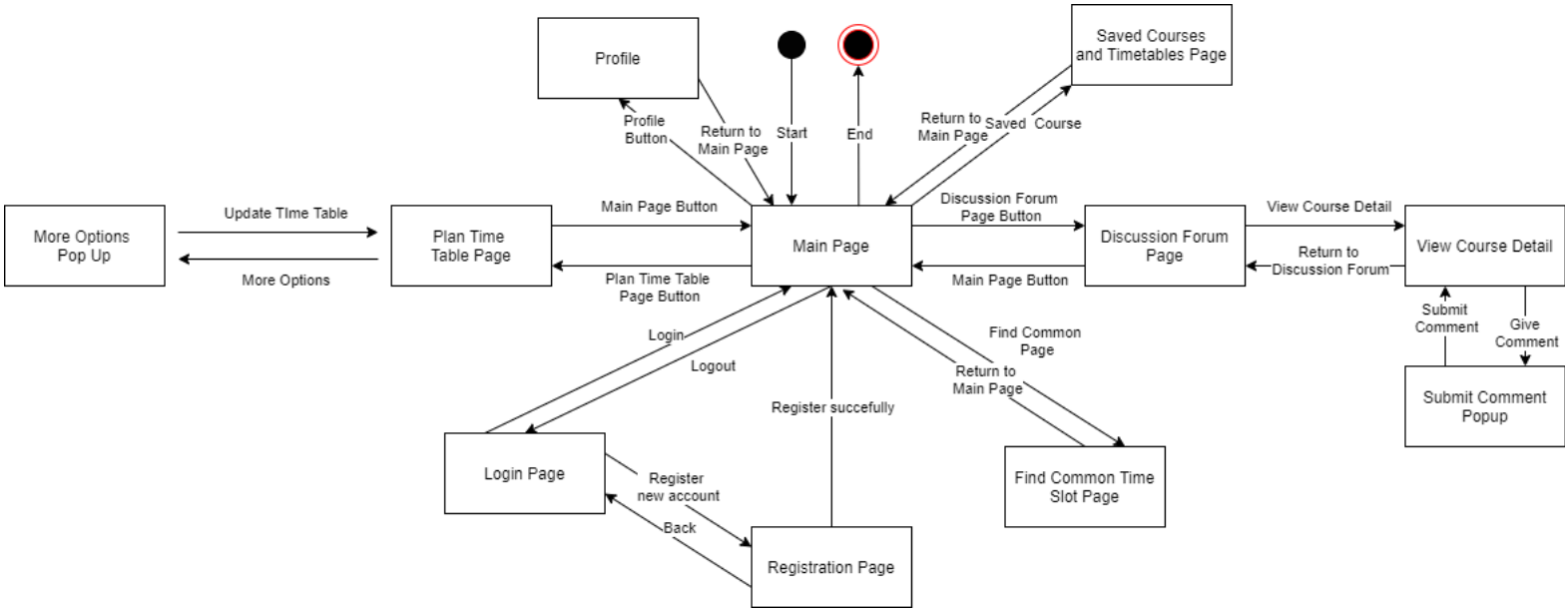


Sequence Diagram for Discussion Forum

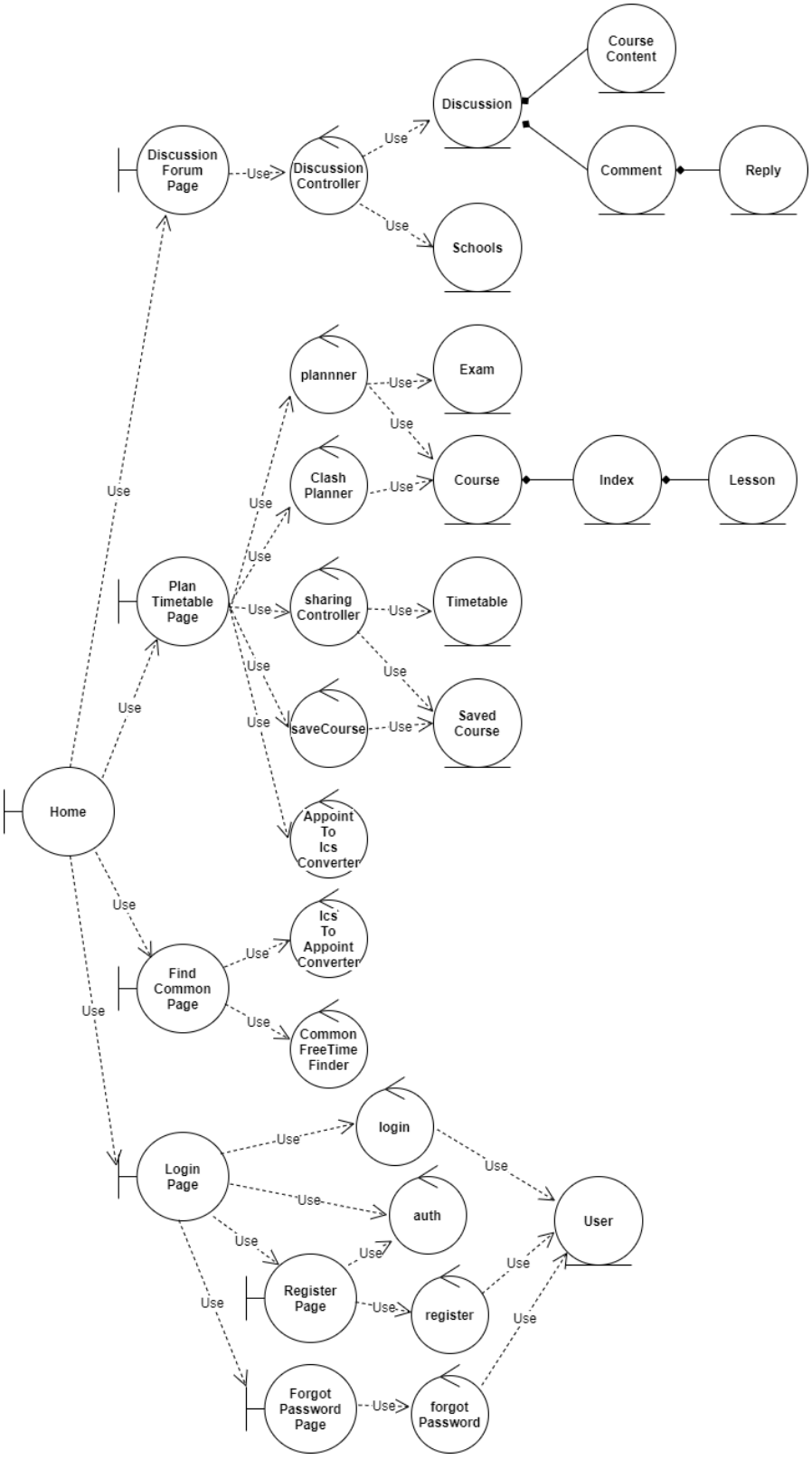
4.8 Use Case Diagram



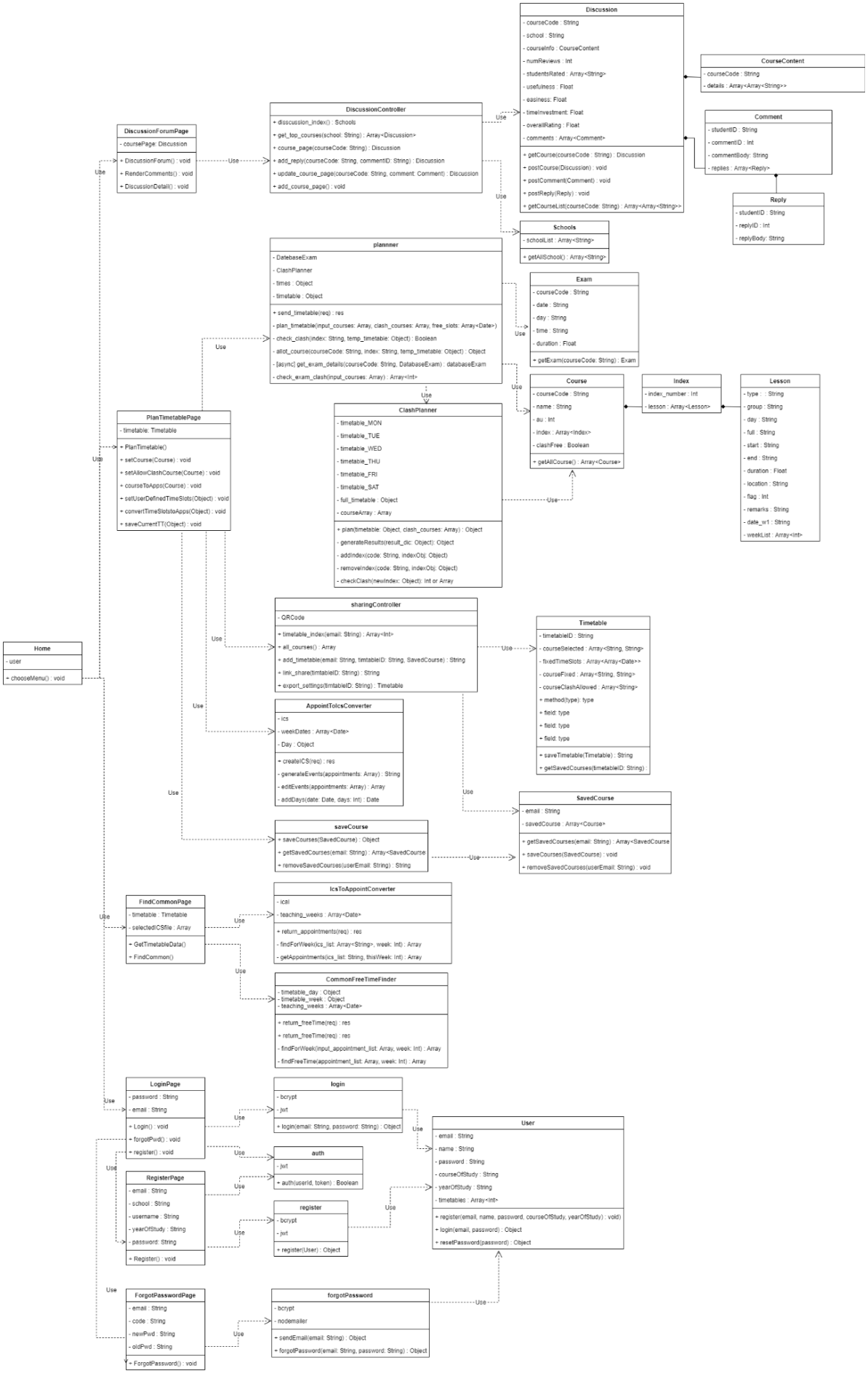
4.9 Dialog Map



4.10 Conceptual Diagram

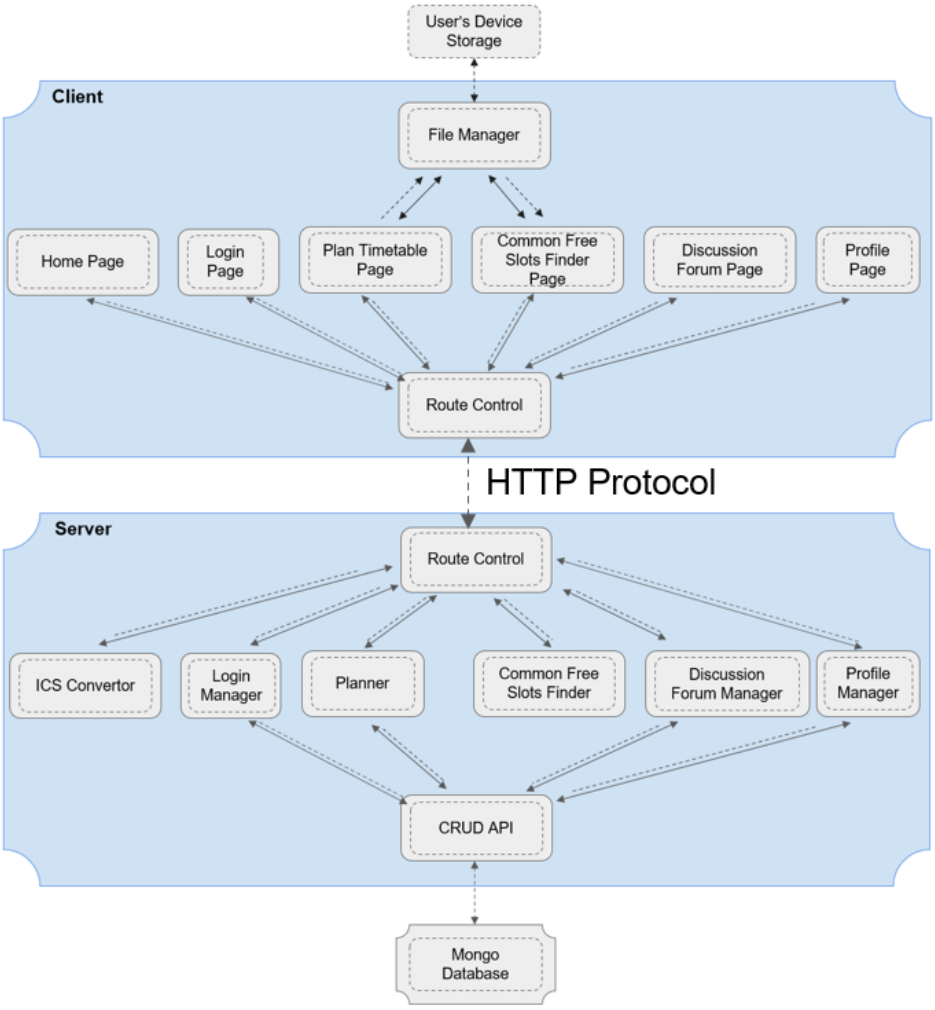


4.11 Class Diagram


Discussion
- courseCode : String
- school : String
- courseInfo : CourseContent
- numReviews : Int
- studentsRated : Array<String>
- usefulness : Float
- easiness: Float
- timeInvestment: Float
- overallRating : Float
- comments : Array<Comment>
+ getCourse(courseCode : String) : Discussion
+ postCourse(Discussion) : void
+ postComment(Comment) : void
+ postReply(Reply) : void
+ getCourseList(courseCode: String) : Array<Array<String>>
CourseContent
- courseCode : String
- details : Array<Array<String>>
Comment
- studentID : String
- commentID : Int
- commentBody: String
- replies : Array<Reply>
Reply
- studentID : String
- replyID : Int
- replyBody: String
DiscussionForumPage
- coursePage: Discussion
+ DiscussionForum() : void
+ RenderComments() : void
+ DiscussionDetail() : void
DiscussionController
+ disscussion_index() : Schools
+ get_top_courses(school: String) : Array<Discussion>
+ course_page(courseCode: String) : Discussion
+ add_reply(courseCode: String, commentID: String) : Discussion
+ update_course_page(courseCode: String, comment: Comment) : Discussion
+ add_course_page() : void
Schools
- schoolList : Array<String>
+ getAllSchool() : Array<String>
plannner
- DatebaseExam
- ClashPlanner
- times : Object
- timetable : Object
+ send_timetable(req) : res
- plan_timetable(input_courses: Array, clash_courses: Array, free_slots: Array<Date>)
- check_clash(index: String, temp_timetable: Object) : Boolean
- allot_course(courseCode: String, index: String, temp_timetable: Object) : Object
- [async] get_exam_details(courseCode: String, DatabaseExam) : databaseExam
- check_exam_clash(input_courses: Array) : Array<Int>
Exam
- courseCode : String
- date : String
- day : String
- time : String
- duration : Float
+ getExam(courseCode: String) : Exam
Course
- courseCode : String
- name : String
- au : Int
- index : Array<Index>
- clashFree : Boolean
+ getAllCourse() : Array<Course>
Index
- index_number : Int
- lesson : Array<Lesson>
Lesson
- type : : String
- group : String
- day : String
- full : String
- start : String
- end : String
- duration : Float
- location : String
- flag : Int
- remarks : String
- date_w1 : String
- weekList : Array<Int>
ClashPlanner
- timetable_MON
- timetable_TUE
- timetable_WED
- timetable_THU
- timetable_FRI
- timetable_SAT
- full_timetable : Object
- courseArray : Array
+ plan(timetable: Object, clash_courses: Array) : Object
- generateResults(result_dic: Object): Object
- addIndex(code: String, indexObj: Object)
- removeIndex(code: String, indexObj: Object)
- checkClash(newIndex: Object): Int or Array
PlanTimetablePage
- timetable: Timetable
+ PlanTimetable()
+ setCourse(Course) : void
+ setAllowClashCourse(Course) : void
+ courseToApps(Course) : void
+ setUserDefinedTimeSlots(Object) : void
+ convertTimeSlotstoApps(Object) : void
+ saveCurrentTT(Object) : void
sharingController
- QRCode
+ timetable_index(email: String) : Array<Int>
+ all_courses() : Array
+ add_timetable(email: String, timtableID: String, SavedCourse) : String
+ link_share(timtableID: String) : String
+ export_settings(timtableID: String) : Timetable
Timetable
- timetableID : String
- courseSelected : Array<String, String>
- fixedTimeSlots : Array<Array<Date>>
- courseFixed : Array<String, String>
- courseClashAllowed : Array<String>
+ method(type): type
+ field: type
+ field: type
+ field: type
+ saveTimetable(Timetable) : String
+ getSavedCourses(timetableID: String)
Home
- user
+ chooseMenu() : void
AppointTolcsConverter
- ics
- weekDates : Array<Date>
- Day : Object
+ createICS(req) : res
- generateEvents(appointments: Array) : String
- editEvents(appointments: Array) : Array
- addDays(date: Date, days: Int) : Date
SavedCourse
- email : String
- savedCourse : Array<Course>
+ getSavedCourses(email: String) : Array<SavedCourse
+ saveCourses(SavedCourse) : void
+ removeSavedCourses(userEmail: String) : void
saveCourse
+ saveCourses(SavedCourse) : Object
+ getSavedCourses(email: String) : Array<SavedCourse
+ removeSavedCourses(userEmail: String) : String
FindCommonPage
- timetable : Timetable
- selectedICSfile : Array
+ GetTimetableData()
+ FindCommon()
IcsToAppointConverter
- ical
- teaching_weeks : Array<Date>
+ return_appointments(req) : res
- findForWeek(ics_list: Array<String>, week: Int) : Array
- getAppointments(ics_list: String, thisWeek: Int) : Array
CommonFreeTimeFinder
- timetable_day : Object
- timetable_week : Object
- teaching_weeks : Array<Date>
+ return_freeTime(req) : res
+ return_freeTime(req) : res
- findForWeek(input_appointment_list: Array, week: Int) : Array
- findFreeTime(appointment_list: Array, week: Int) : Array
LoginPage
- password : String
- email : String
+ Login() : void
+ forgotPwd() : void
+ register() : void
login
- bcrypt
- jwt
+ login(email: String, password: String) : Object
auth
- jwt
+ auth(userId, token) : Boolean
User
- email : String
- name : String
- password : String
- courseOfStudy : String
- yearOfStudy : String
- timetables : Array<Int>
+ register(email, name, password, courseOfStudy, yearOfStudy) : void)
+ login(email, password) : Object
+ resetPassword(password) : Object
RegisterPage
- email : String
- school : String
- username : String
- yearOfStudy : String
- password: String
+ Register() : void
register
- bcrypt
- jwt
+ register(User) : Object
ForgotPasswordPage
- email : String
- code : String
- newPwd : String
- oldPwd : String
+ ForgotPassword() : void
forgotPassword
- bcrypt
- nodemailer
+ sendEmail(email: String) : Object
+ forgotPassword(email: String, password: String) : Object
Use

5. System Architecture

5.1 Architecture Diagram



5.2 Explanation for Client-Server Architecture

The “MyCal” web application is constructed by frontend, which was coded by react and node, and backend, which was coded by express and node. The frontend is the client and the backend is the server.

At the frontend, the page controllers control the layout for each web page and receive user input. The route control will send the requests for data processing, retrieving and updating through the right url route. The file manager handles uploading and downloading events between the web page and the user’s device.

As for the backend, the route control receives the requests from the frontend and passes data and control to the correct manager, which contains the logic to process the data. Some

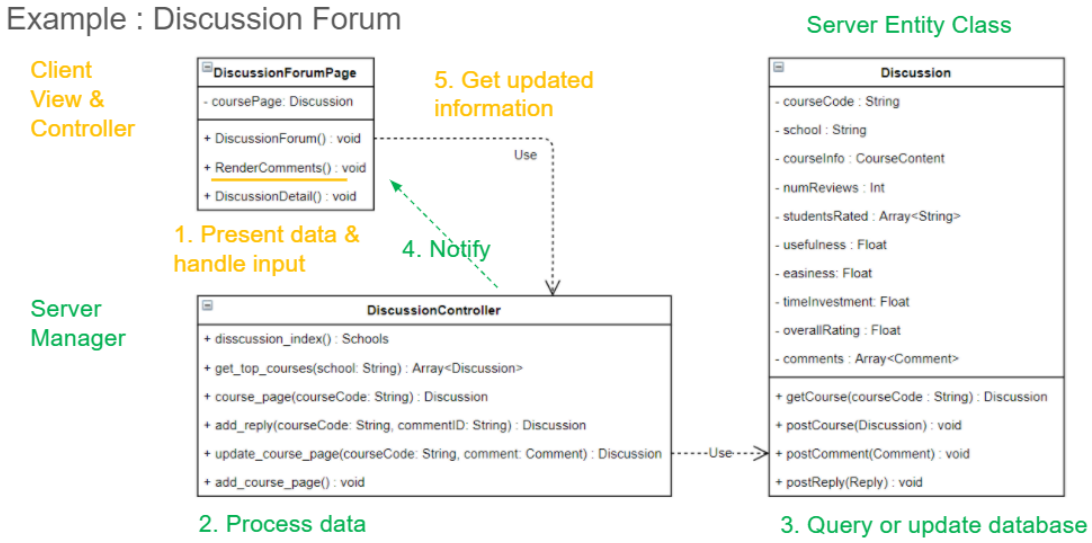
managers have the access to the mongo database through the CRUD (Create, Remove, Update and Delete) API.

The frontend's request and the backend's response are sent through HTTP protocol.

By adopting the Client-Server Architecture, data is centralized, making the system easy to maintain. The frontend can be run on multiple client devices and does not require much complex computation, while the centralized backend server can handle all the requests from the clients, perform complex computation, such as generating hundreds of course index combinations, and send the resources to clients easily. In this process, the app logic is also hidden within the backend, keeping the clients from accessing it.

5.3 Design Patterns

5.3.1 Observer Pattern Example



In the discussion forum, after the user creates a new comment, it will be immediately displayed in the course page. In this process, the client is the observer and the server is the subject.

The ForumPage Class at the client side is the View and Controller, which presents the data and handles user interaction. The server's Manager and Entity classes serve as the subject, which encapsulates the logic to process user input and query or update the database.

After updating is complete, the DiscussionController Class will return a value to the ForumPage Class to notify it to use RenderComment() method to request for the updated comment list.

In the future, more views could be added to the course page in the discussion forum. For example, a view showing a short list of comments which get the most upvotes. By adding another method RenderTopComment() to the ForumPage Class, 2 views can be updated simultaneously after new comments are added or comments get upvoted. Therefore, this design

pattern can be easily expanded to Model-View-Controller (MVC) Architecture to allow this website to be more interactive.

6. Other Nonfunctional Requirements

6.1 Usability Requirements

6.1.1 The system must allow easy reversal of the user's actions.
- 6.1.1.1 The user must be able to remove the courses added by clicking on the remove course button and also remove previously saved courses and timetables

6.1.2 The system must have consistency
- 6.1.2.1 A consistent sequence of actions is required for similar situations
- 6.1.2.2 A consistent visual layout matching the website's theme should be adopted(eg. label, fonts and color palette)

6.1.3 The system must provide informative feedback
- 6.1.3.1 To provide necessary feedback to the user when invalid inputs are detected
- 6.1.3.2 To display an appropriate error message when an exception occurs
- 6.1.3.3 To display an appropriate warning message when the user is about to undertake an irreversible process.

6.2 Security Requirements

6.2.1 The system will only store hashed passwords into the database to prevent hacking attempts.
6.2.2 On login, the system will compare the salt-hashed version of the user password with the hashed password stored in the database and will allow access only when the two match.
6.2.3 The system will mask the password field with filled disks to prevent onlookers from stealing the password.
6.2.4 The system will store customer data. Only the user himself/herself will be able to edit the data. The data will not be used for purposes other than storage under PDPA.

6.3 Performance Requirements

6.3.1 The system must be able to generate and display at least 100 timetables(if the course combinations permit) within 4 seconds.
6.3.2 The system must not crash at any process during the website's lifetime.
6.3.3 The system must be able to run with little or no downtime.
- 6.3.3.1 Any maintenance work should be done on a different branch of the codebase to allow users to continue using the main branch.

6.3.4 The system must have fast response time (within 50 milliseconds) in order to prevent the user from experiencing any input lag or latency issues.

7. Testing

7.1 Black Box Testing

1. Login

a. Generic cases

Test Id	Scenario	Expected Result	Actual Result
1	Login with valid email and password	The system displays the homepage.	The system displays the homepage.
2	Login without filling up the required fields	The system prompts the user to fill up the required fields for logging in.	The system prompts the user to fill up the required fields for logging in.
3	Login with invalid email and valid password	The system prompts users to enter email and password again.	The system prompts users to enter email and password again.
4	Login with valid email and invalid password	The system prompts users to enter email and password again.	The system prompts users to enter email and password again.

b. Specific Cases (Combination)

Email	Password	Expected Result	Actual Result
asthagarg1611@gmail.com	MyAccount	Successful Login	Successful Login
wrong@gmail.com	ntuadmin123	Invalid email/password	Invalid email/password
testing@gmail.com	wrongPass	Invalid email/password	Invalid email/password
Empty	ntuadmin123	Please enter valid email	Please enter valid email
test1@ntu.edu.sg	Empty	Please enter password	Please enter password
Empty	Empty	Please enter valid email, Please enter password	Please enter valid email, Please enter password

2. **Equivalence class and boundary value testing for Generating Common Free Time Slots**

- Number of timetables for valid equivalence class (1<= x <= 10)
    - Lower Boundary : 0, **1**, 3
    - Upper Boundary : 9, **10**, 11

Hence:

- Valid Boundary Values: {1, 10}
- Invalid Boundary Values: {0, 11}

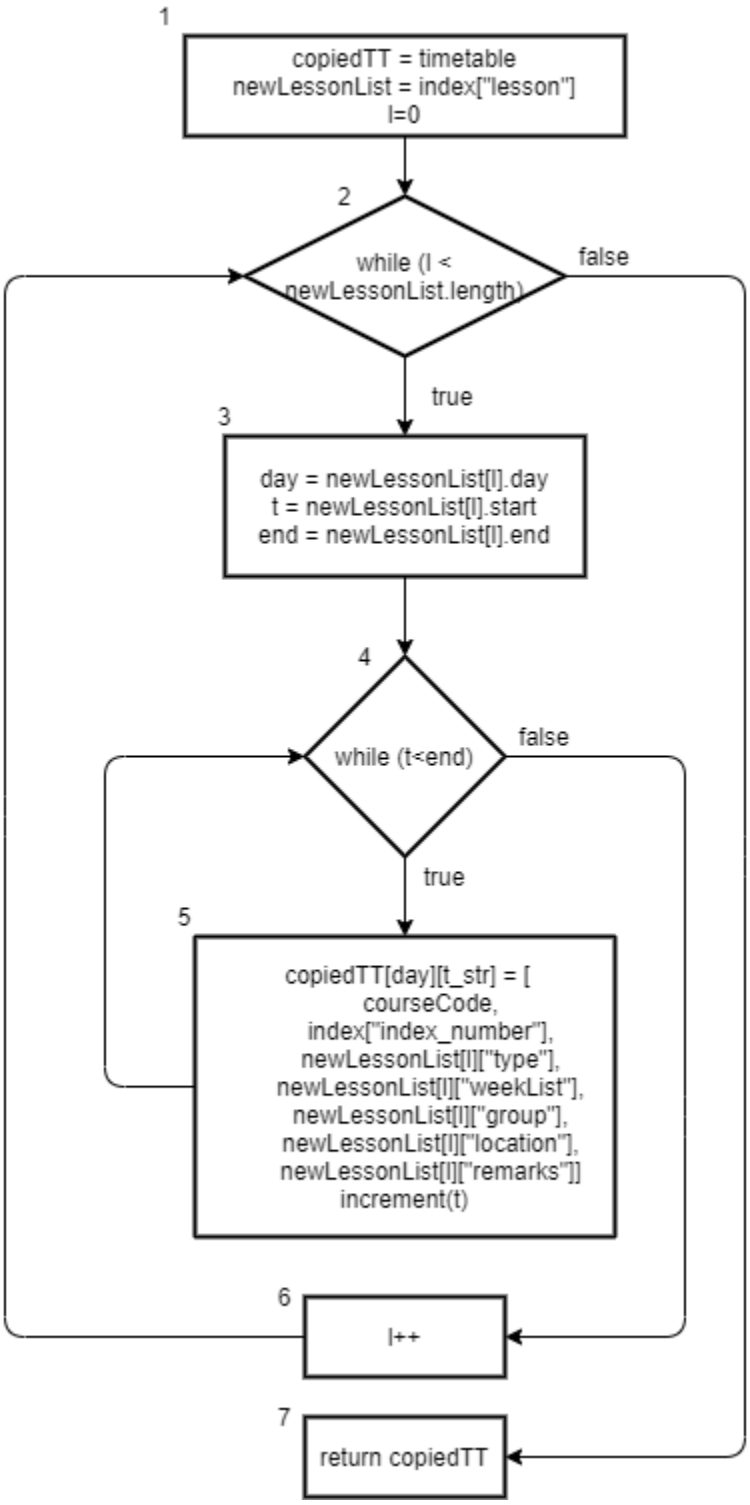
TestID	Number of Timetables uploaded	Expected Result	Actual Result
1.	0	The system displays an alert to upload another timetable.	The system displays an alert to upload another timetable.
2.	1	The common free time slots of the single timetable should be displayed.	The common free time slots of the single timetable are displayed.
3.	10	The common free time slots of all the timetables should be displayed.	The common free time slots of all the timetables are displayed.
4.	11	The system should show an alert that the number of timetables have exceeded the set limit.	The system shows an alert that the number of timetables have exceeded the set limit.

7.2 White Box Testing

1. White box testing for allott_course for the Planner controller

The function adds lessons of a specific index of a course to the timetable after it has been verified that those lessons do not clash with any existing lessons (or if they do then it has been allowed by the user). The function receives the timetable which has been planned till that moment and the list of lessons which are to be added containing the information about the

starting time, ending time, location, type of lesson and the weeks in which it will be conducted. It returns the timetable with the lessons added to the received timetable.



Control Flow Diagram for the allot_course function

Cyclomatic Complexity = |decisionpoint| + 1 = 2 + 1 = 3
Set of basis paths:
i) 1, 2, 7
ii) 1, 2, 3, 4, 6
iii) 1, 2, 3, 4, 5, 6, 7

Test cases:
i) l = 1, newLessonList.length=1
ii) l = 0, newLessonList.length=1, t=1230, end = 1230
iii) l = 0, newLessonList.length=2, t=(1030,1130), end = (1100, 1200)

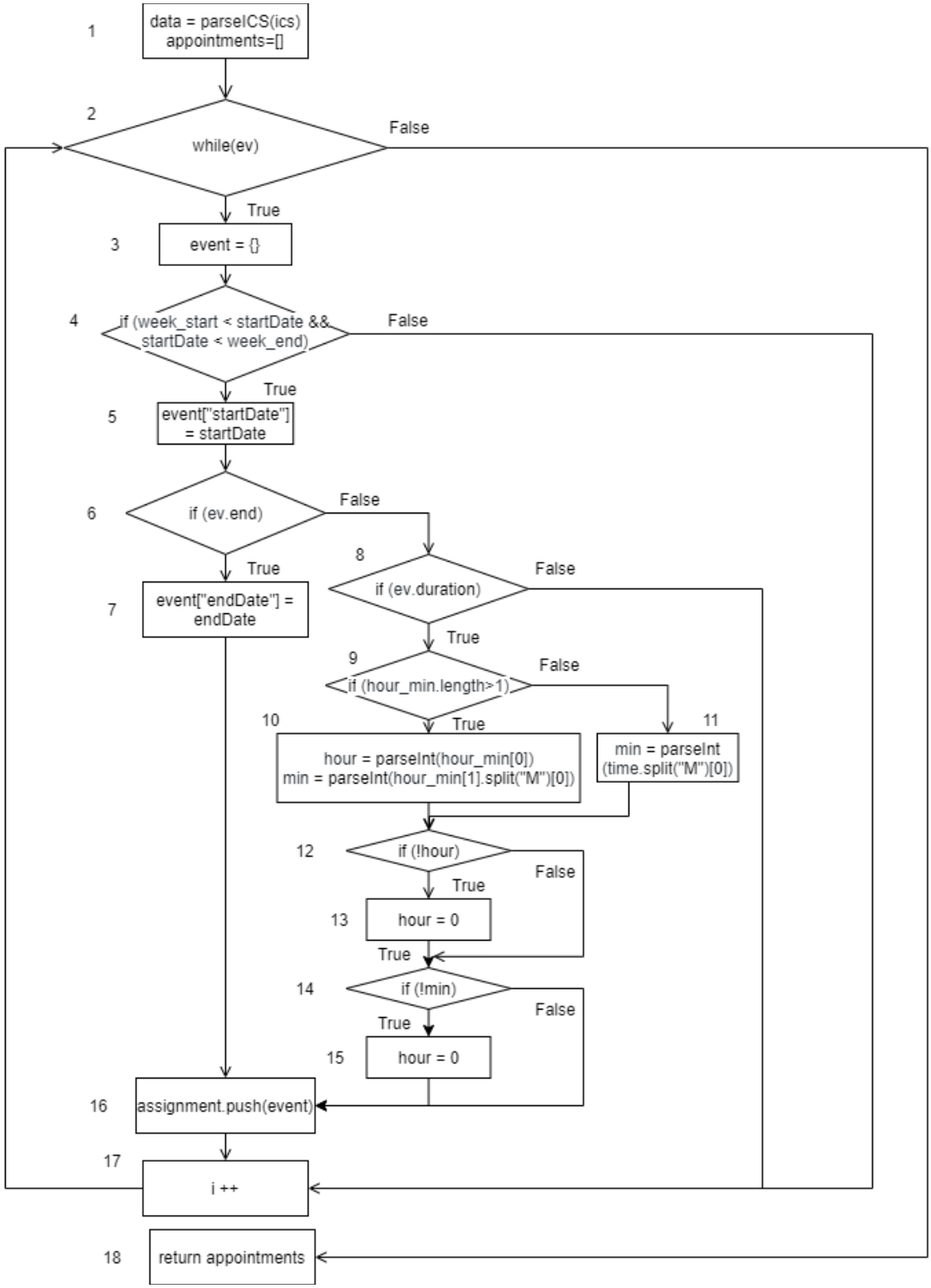
Real execution paths:
i) 1, 2, 7
ii) 1, 2, 3, 4, 6, 2, 7
iii) 1, 2, 3, 4, 5, 4, 6, 2, 3, 4, 5, 4, 6, 2, 7

No.	l	t	End	Expected result	Actual result
1	1	-	-	copiedTT is returned with the previous lessons added.	copiedTT is returned with the previous lessons added.
2	1	1230	1230	copiedTT is returned with the previous lessons added.	copiedTT is returned with the previous lessons added.
3	2	(1030, 1130)	(1100, 1200)	2 lessons with the corresponding starting time and ending time are added to copiedTT which is then returned.	2 lessons with the corresponding starting time and ending time are added to copiedTT which is then returned.

Note: In the function it is not possible to have newLessonList.length to be 0 as there will be at least 1 lesson for every index, and index is added to the timetable only if there is no clash with other lessons.

2. White box testing for getAppointments() method in IcsToAppointConverter Class

This function extracts all the events during the target week from an ICS string. The inputs for this function are ***ics***, the ICS string, and ***week***, the target week number in teaching weeks (1~13). The output of this function is an array of event dictionaries. White box testing ensures that this function will not return any event dictionary that has no “endDate” which the frontend could not recognize.



Cyclomatic Complexity = |decisionpoint| + 1 = 7 + 1 = 8

Set of basis paths:

i) 1, 2, 15

ii) 1, 2, 3, 4, 5, 6, 7, 14, 2, 15

iii) 1, 2, 3, 4, 14, 2, 15
iv) 1, 2, 3, 4, 5, 6, 8, 14, 2, 15
v) 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 14, 2, 15
vi) 1, 2, 3, 4, 5, 6, 8, 9, 13, 11, 12, 14, 2, 15
vii) 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 14, 2, 15

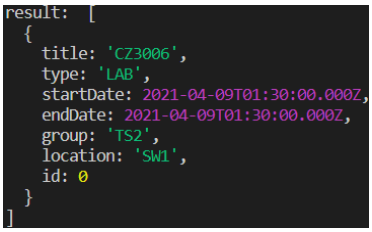
For testing, the basic format of the ICS string was set to be:

```
BEGIN:VCALENDAR
VERSION:2.0
CALSCALE:GREGORIAN
PRODID:adamgibbons/ics
METHOD:PUBLISH
X-PUBLISHED-TTL:PT1H
BEGIN:VEVENT
UID:ab4e7a6e-db0a-4980-b192-1fb953d1772f
SUMMARY:CZ3006 LAB
DTSTAMP:20210410T070200Z
DTSTART:20210409T013000Z
DTEND:20210409T033000Z
DURATION:PT1H30M
DESCRIPTION:TS2
LOCATION:SW1
END:VEVENT
END:VCALENDAR
```

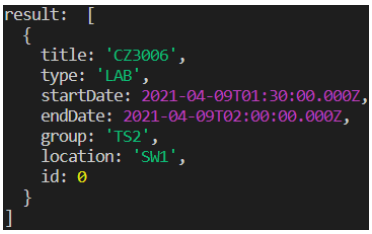
This ICS string contains an event called “CZ3006 LAB” taking place from 1:30am to 3:30am (UTC time) on April 9th, 2021. The duration is 1.5 hours. This event is on week 12 according to NTU’s academic calendar.

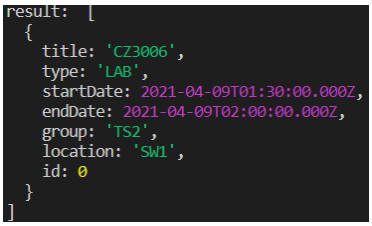
During testing, only **DTEND**, **DURATION** and **week** will be changed.

Test cases:
i) no event, week 12
ii) DTEND: 20210409T033000Z, DURATION: - , week 12
iii) DTEND: 20210409T033000Z, DURATION: - , week 13
iv) DEND: - , DURATION: - , week 12
v) DEND: - , DURATION: PTHM, week 12
vi) DTEND:-, DURATION: PT1HM, week 12
vii) DTEND: -, DURATION: PTH30M, week 12
viii) DTEND: -, DURATION: PT30M, week 12

No.	DEND	DURATION	week	Expected result	Actual result
1	No event in the ICS string.		12	[]	result: []
2	20210409T033000Z	-	12	[{ title: 'CZ3006', type: 'LAB', startDate: 2021-04-09T01:30:00.000Z, endDate: 2021-04-09T03:30:00.000Z, group: 'TS2', location: 'SW1', id: 0 }]	
3	20210409T033000Z	-	13	[]	result: []
4	-	-	12	[]	result: []
5	-	PTHM	12	[{ title: 'CZ3006', type: 'LAB', startDate: 2021-04-09T01:30:00.000Z, endDate: 2021-04-09T01:30:00.000Z, group: 'TS2', location: 'SW1', id: 0 }]	
6	-	PT1HM	12	[{ title: 'CZ3006', type: 'LAB', startDate: 2021-04-09T01:30:00.000Z, endDate: 2021-04-09T02:30:00.000Z, group: 'TS2', location: 'SW1', id: 0 }	

				]	

7	-	PTH30M	12	[
				{
				title: 'CZ3006',
				type: 'LAB',
				startDate:
				2021-04-09T01:30:00.000Z,
				endDate:
				2021-04-09T02:00:00.000Z,
				group: 'TS2',
				location: 'SW1',
				id: 0
				}
				]
					

8	-	PT30M	12	[
				{
				title: 'CZ3006',
				type: 'LAB',
				startDate:
				2021-04-09T01:30:00.000Z,
				endDate:
				2021-04-09T02:00:00.000Z,
				group: 'TS2',
				location: 'SW1',
				id: 0
				}
				]
					

Appendix A: Glossary

Data Dictionary

Word	Description
Timetable	Table showing the overall schedule of the user including class schedule and personal schedule are shown.
Module/Course	A unit of teaching which lasts one academic term which is led by one or more Instructors. A module may contain lecture, tutorial, lab and seminar.
Course code	A unique code which is used to identify a course. Each course has only one course code.
Index	An integer used together with the course code to uniquely identify the exact timing of the course schedule. A course must have at least one index.
Class	The umbrella term for the teaching activities, including lectures, tutorials, labs, seminars and physical training, which a course may have.
Discussion forum	A platform used by students to provide feedback regarding courses in university.
Assessment type	A variety of methods used by course instructors to gauge the learning progress and skills acquisition of students from a certain course. Assessment types include continuous assessments, projects and final examinations.
Grading format	A rubric which is used to grade students' works based on some specific benchmarks.
Academic Unit/AU	A single digit integer that maps the time investment in number of hours required per week required by the course