Software Design Document

for

Virtual Room Reservation Assistant

Prepared by:

1. **B10504028 林哲豪**
2. **B10601120 許世佑**
3. **B10632011 陳彥瑜**
4. **B10632037 潘莫同**

CS3025301 Software Engineering

2020/12/13

Table of Content

Revision History

Date	Version	Description	Author
17/12/2020	0.1	Creation	B10504028 林哲豪
24/12/2020	1.0	reformat	B10504028 林哲豪
06/01/2021	1.1	Google reminder / Change password / Database description change GUI picture change	B10632011 陳彥瑜 B10504028 林哲豪

1. Introduction

1.1 Purpose

This architecture document aims to provide our developer a detailed view of how the system was architect and also make the discussion of architecture design more clear. Also show all of the stakeholders how all of the functional and nonfunctional requirements are implemented.

1.2 Scope

This virtual room reservation assistant software aims to provide users with a butter-smooth experience using it. For the best experience and efficient developing process we separate the frontend app and backend service apart, so the developers of backend service don't bother to handle html, css and javascript at all, and frontend GUI developer don't have to deal with database and python.

1.3 Overview

Architecture of backend services and all of the subsystem are documented in chapter 3,4 and 5, architecture of frontend GUI are documented in chapter 6, external reference and document related to development are listed in the following section and chapter 8.

1.4 Reference Material

Rafael Kaliski(2020), Software Design Document Template, National Taiwan University of Science and Technology

1.5 Definition and Acronyms

Term	Definition
RESTful API	Application programming Interface that follows the Software architectural style that defines a set of constraints to be used for creating web service.
Vue.js	Progressive frontend application framework that provides a rich set of data binding functions, and UI component reusing.
Vue CLI	A set of services for compiling, building, serving and style checking a frontend app written in Vue.js.
Event Listener	Software observer function for monitoring the state change of UI components or system services.

Event Handler	Software callback function triggered by an event listener to respond to the action of user or state change of system services.
SQLite	Lightweight database that integrated into software could be accessed with API, unlike full-featured RDBM, SQLite is not an independent process.

2. System Overview

In this section, we would go through the entire system from surface to the bottom of the deep ocean, since this system is built on the base of many advanced technologies.

2.1 System Functionality

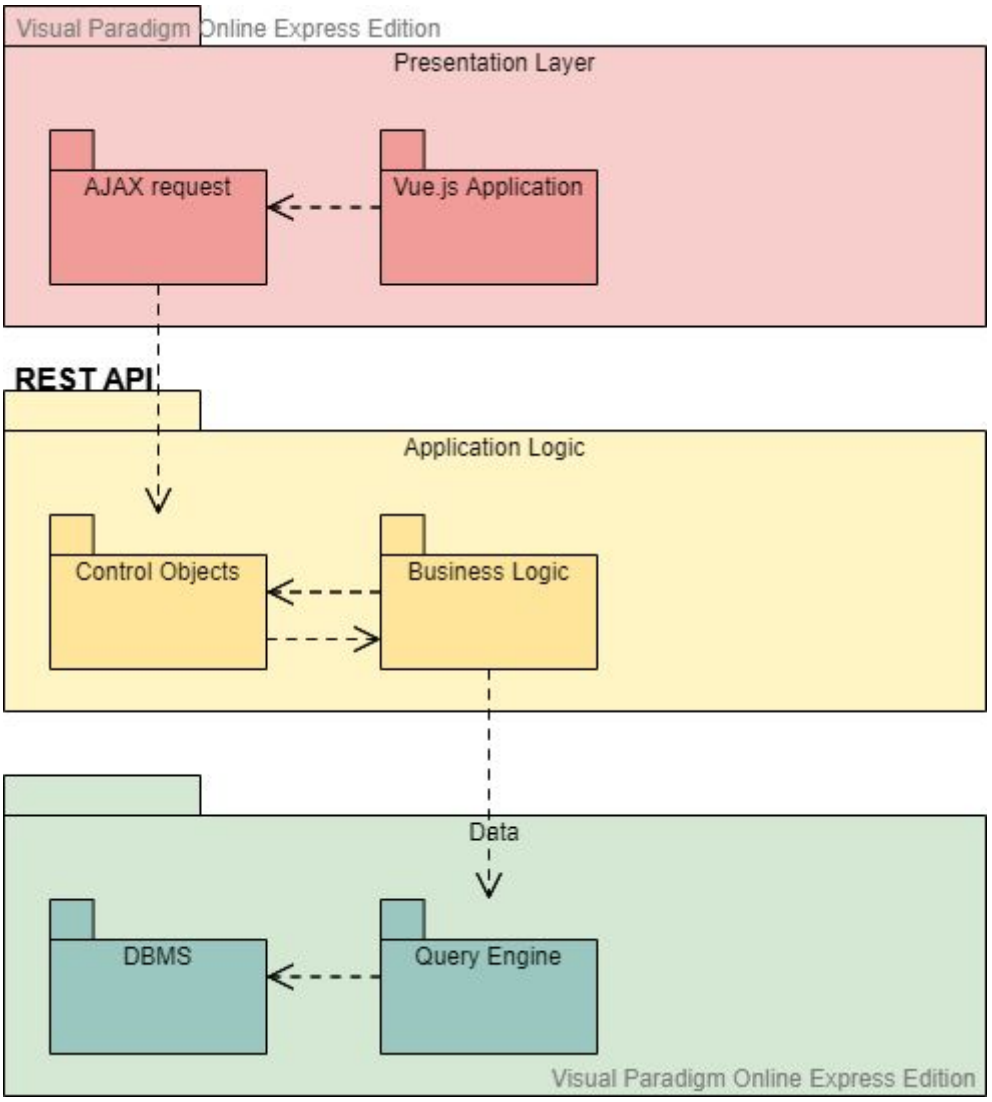
As stated before, this system helps users to reserve a meeting room and create a meeting reminder to the attendee, all of the meeting events are editable. Users could see the occupation of each meeting room and the event per month. This system provides two ways of accessing the service, one is from our customized GUI and the other with web API. Developers may want to embed this service in their software and the web API is an available way to go.

2.2 System Context

Under the hood, this system decomposed to several subsystems, customized GUI server and web API server. On GUI server, we use Single Page Application(SPA) solution, which delivers the whole APP at once, all of the event handlers, data structure, components and different pages are included in the APP. Of the backend service, we use Django as our backend framework, Django has a lot of extra features like database management, user authentication.

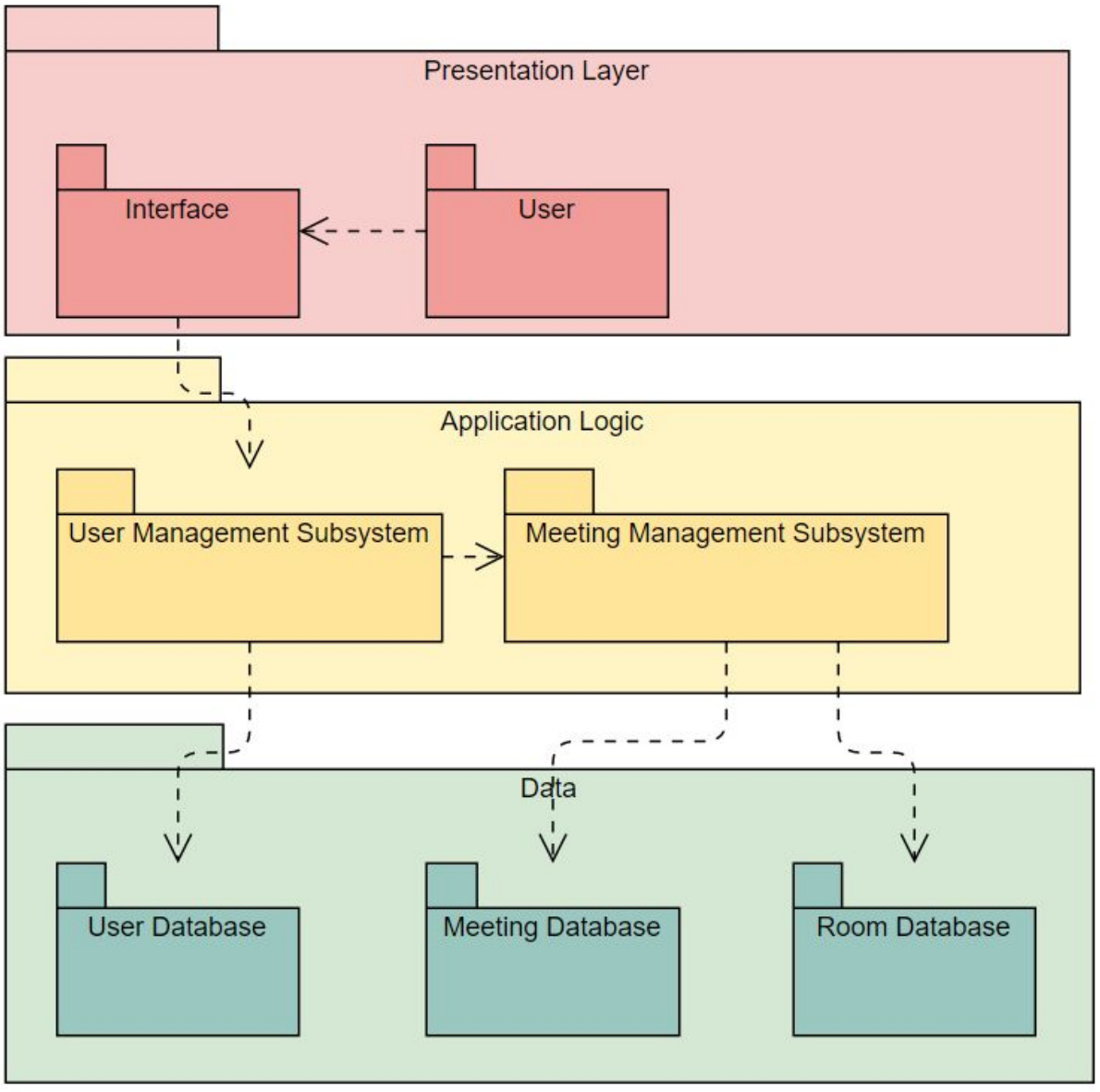
2.3 System Design

This system is designed based on MVC pattern, which decouple the data model, ui view and program controller. The communication between different layers of the system is based on mature interfaces like REST API, and ORM.


Visual Paradigm Online Express Edition
Presentation Layer
AJAX request
Vue.js Application
REST API
Application Logic
Control Objects
Business Logic
Data
DBMS
Query Engine
Visual Paradigm Online Express Edition

3. System Architecture

3.1 Architecture Design

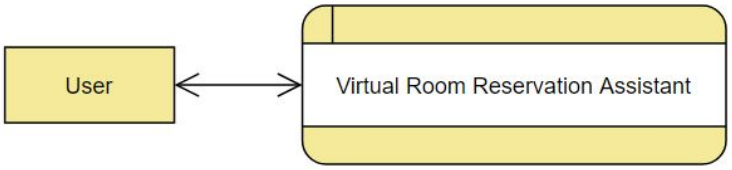


- Interface : provides interaction between system and user, rendering content of user and database as well.
- User Management Subsystem : provides the facilities that covers all the user management functionalities.
- Meeting Management Subsystem : involves the main operations on the Meeting object.
- Database : We use sqlite as our database. It provided data insertion, deletion, advise, search.

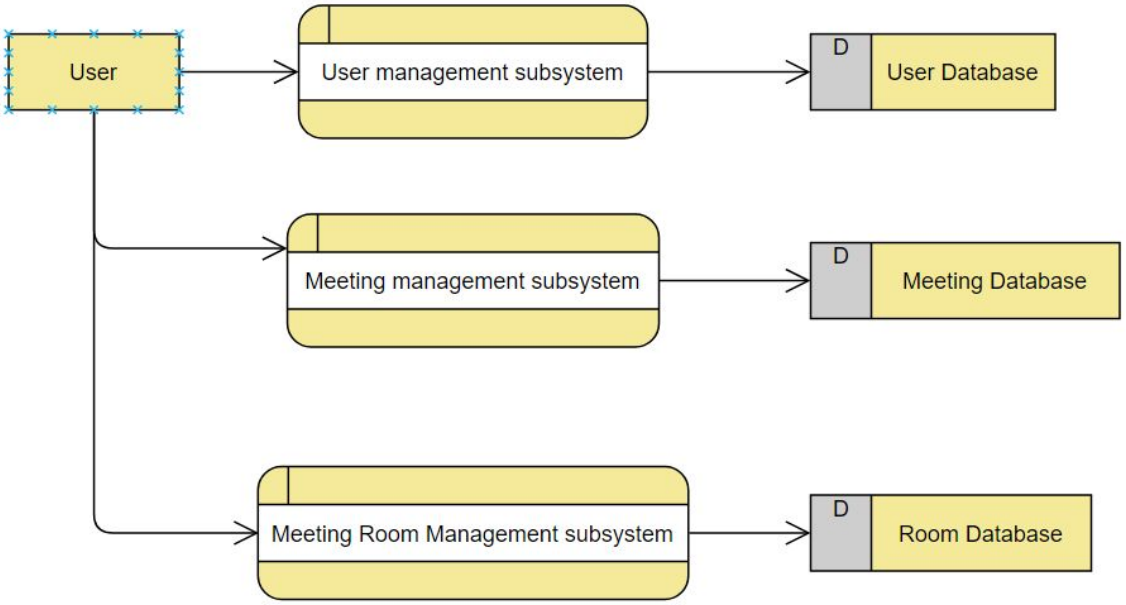
3.2 Decomposition Description

3.2.1 Data Flow Diagram

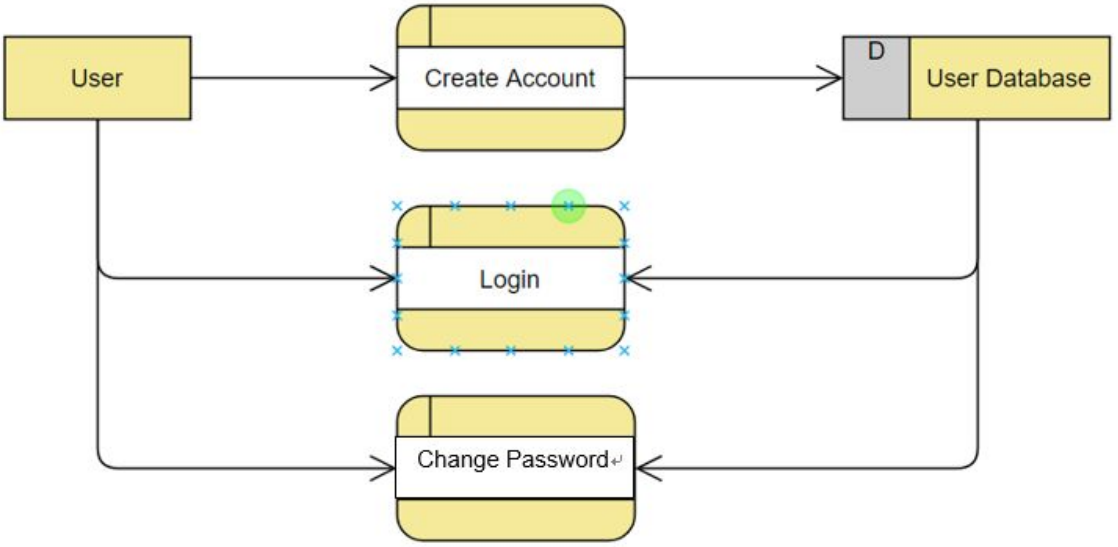
layer0 (background) :

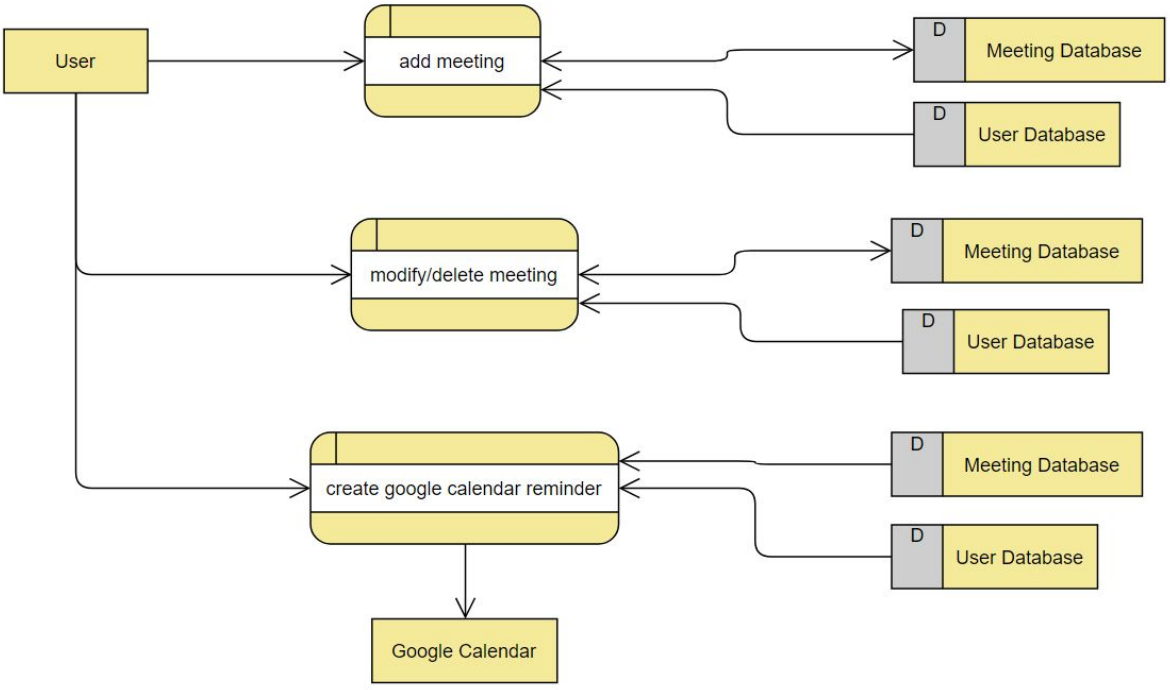


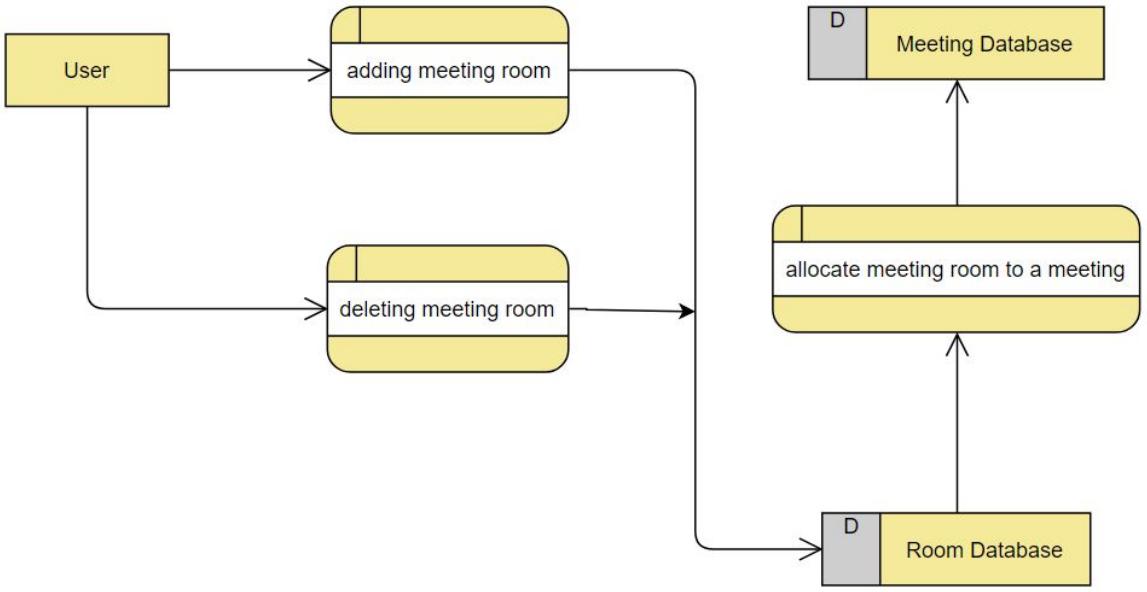
layer1 :



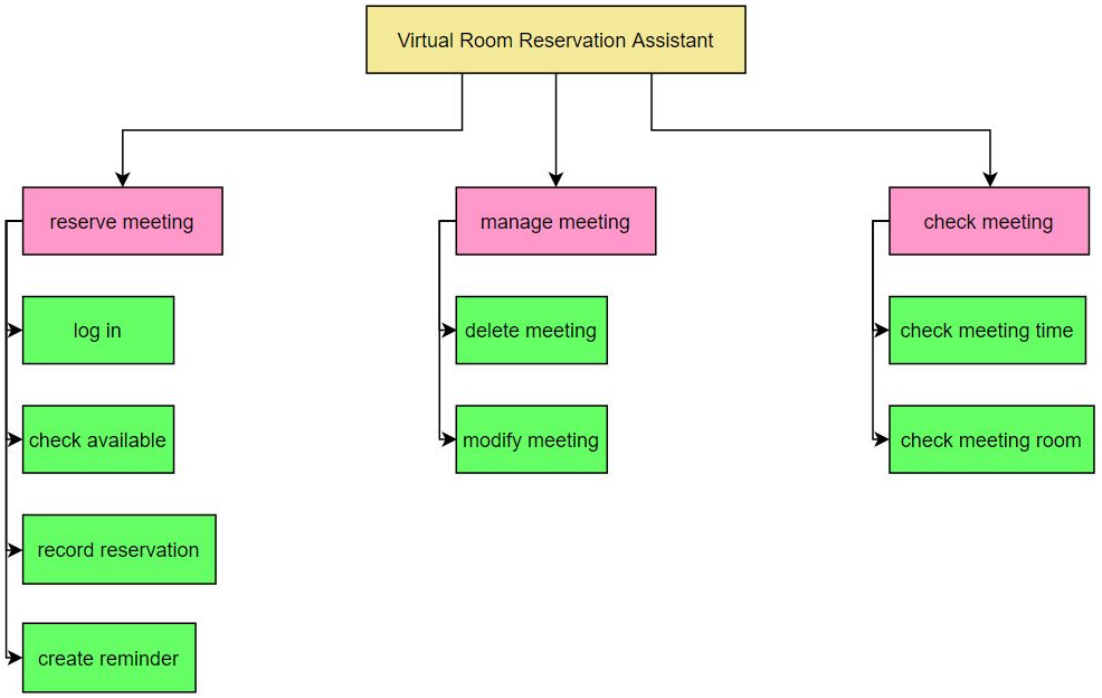
layer2 :


User
Create Account
D
User Database
Login
Change Password


User
add meeting
D
Meeting Database
D
User Database
modify/delete meeting
D
Meeting Database
D
User Database
create google calendar reminder
D
Meeting Database
D
User Database
Google Calendar



3.2.2 Structural Decomposition Diagrams



3.3 Design Rationale

1. Able to display clearly, and classify user authority .

2. Make collaboration incredibly easy.

3. Select SQLite as our database so that we can use the language we are familiar with, and it supports synchronization between user end and database, no need to save data in the local host.

4. Data Design

4.1 Data Description

There is one database in this project, which contains three tables. The first is the User Table, which is maintained by django.contrib.auth.models. The second is the Meeting Table. The third is the Room Table. We use SQLite as our database.

4.2 Data Dictionary

4.2.1. User Table

Field	**Type**	**Description**
Name	Text	Primary key.
Mail	Text	The mail address of this user.
Account	Text	The account name of the user.
Password	Text	The password of the account.

4.2.2. Meeting Table

Field	Type	Description
Topic	Text	Primary key.
Host	Text	Host's name of the meeting
Start	Text	Start time of the meeting.
End	Text	End time of the meeting.
RoomID	Text	Indicate where the meeting will be held. Foreign key.
Attendee	List	List the attendee's name.

4.2.3. Room Table

Field	Type	Description
RoomID	Text	Primary key.
Date	Text	The date of meeting(if any.)

5.Component

5.1 LoginPage

- login : send user input to database for authentication.
- register : register a new account to database

5.2 Reserve meeting

- meeting_create(room_id, topic, host, start, end, attendee, location)
- google_reminder_create(meeting_id, reminder_time)

5.3 Manage meeting

- meeting_edit(task_id,room_id, topic, host, start, end, attendee, location)
- meeting_cancel(task_id, room_id, attendee)

5.4 Check meeting

- get_date_info() : update the latest occupied time for each month.
- get_room_info() : update the latest occupied time for each room.

6.Human Interface Design

6.1 Overview of User Interface

6.1.1 Technology Explain

In this project we use a different approach of web UI. Traditionally, the web page would be rendered on the server using a template engine like Jinja2 on Flask. But this approach couples the content and UI together and makes it difficult for UI developers to design whatever UI they want. So, we use a technology called Single Page Application(SPA), which the entire application UI is delivered all at once, and using Javascript to manipulate DOM to make AJAX requests and change the content on screen.

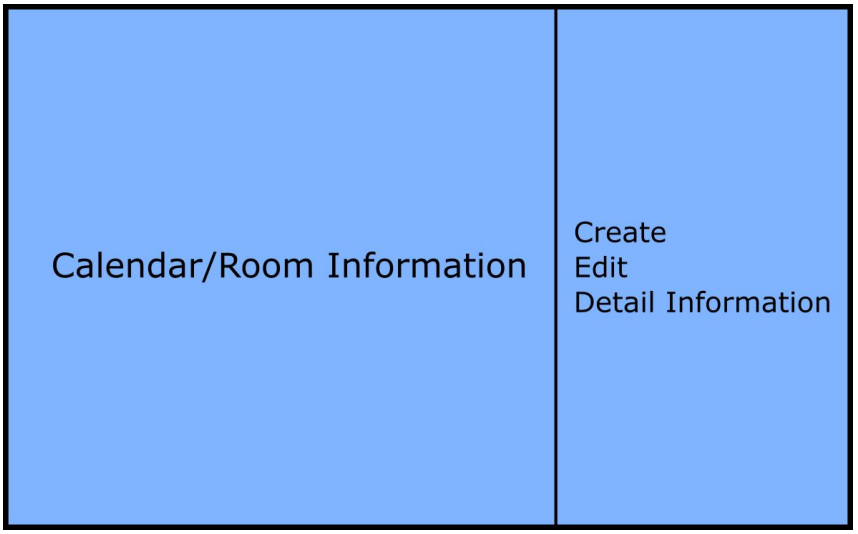
6.1.2 Functionality Explain

Users could visit our website with browser, the initial page would be login page, or signup page, user enter their personal information and later could use the account and password to get the authorization of this application. After logged in, the user would see the main page where we put all of the main functionality in, including an event calendar, meeting

information and other functionalities could be accessed with the related label on it. Room information could be accessed by press the room information button, create meeting by press create meeting button and editing meeting information by press edit button. Detailed meeting information showed on the side of the page.

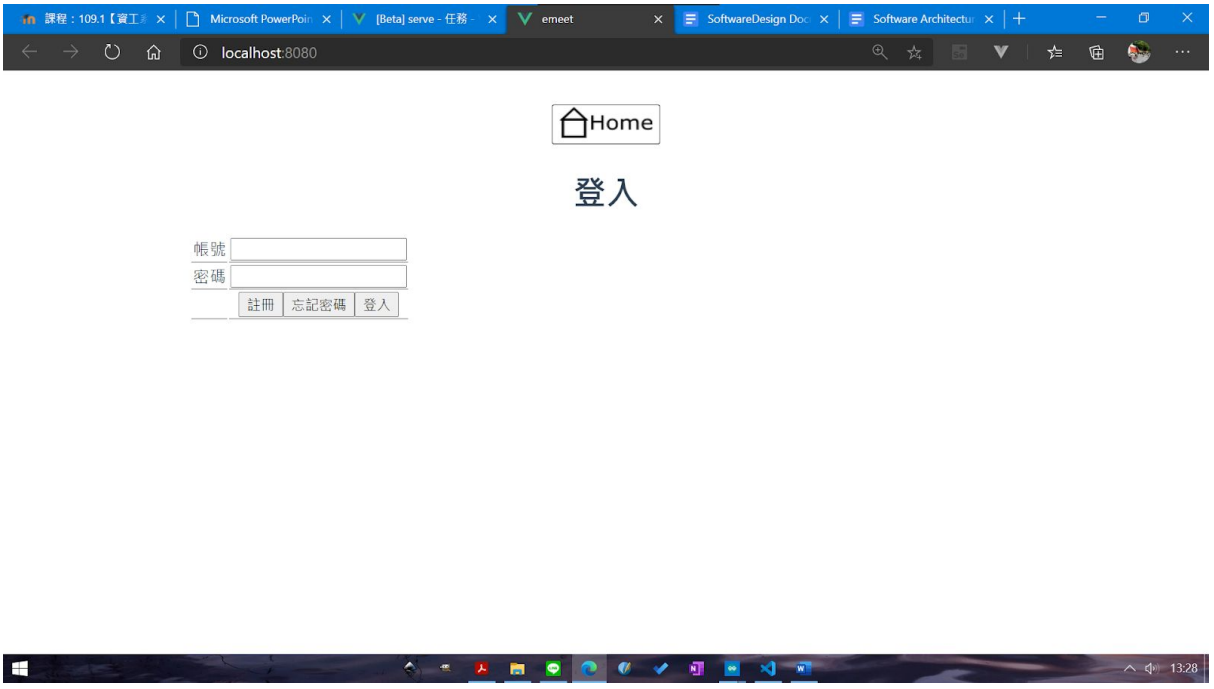
6.1.3 Basic UI Layout

The main page was divided into two parts, the left part is calendar and room information with simple meeting information in them. The right part is responsible for detailed information and editing meeting information.
This left and right column design makes the user focus on a single page instead of jumping around pages. Also, it makes the document structure and UI component structure much simpler.

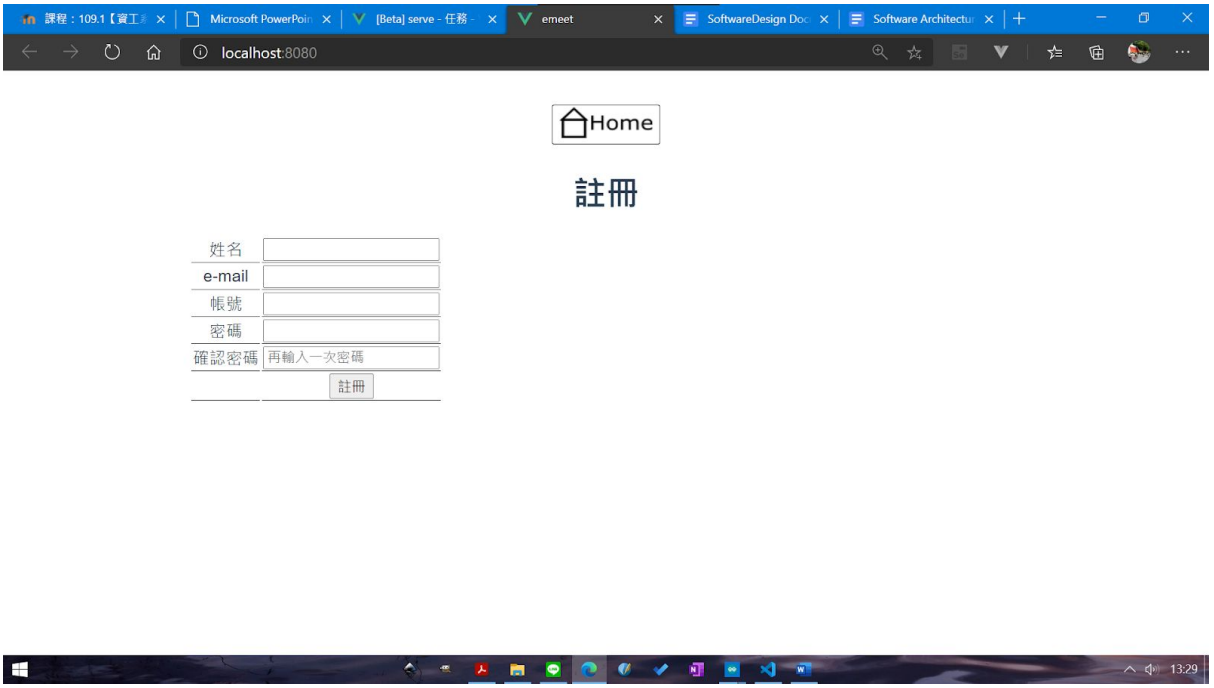


6.2 Screen Images

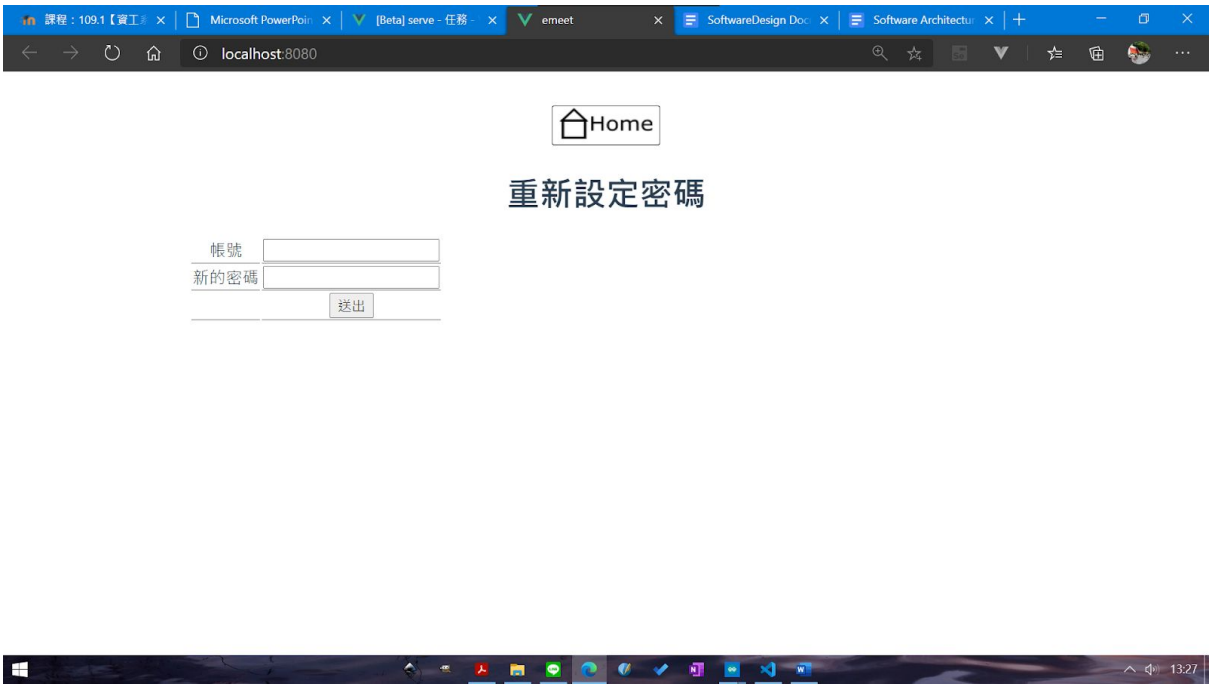
6.2.1 Login Page



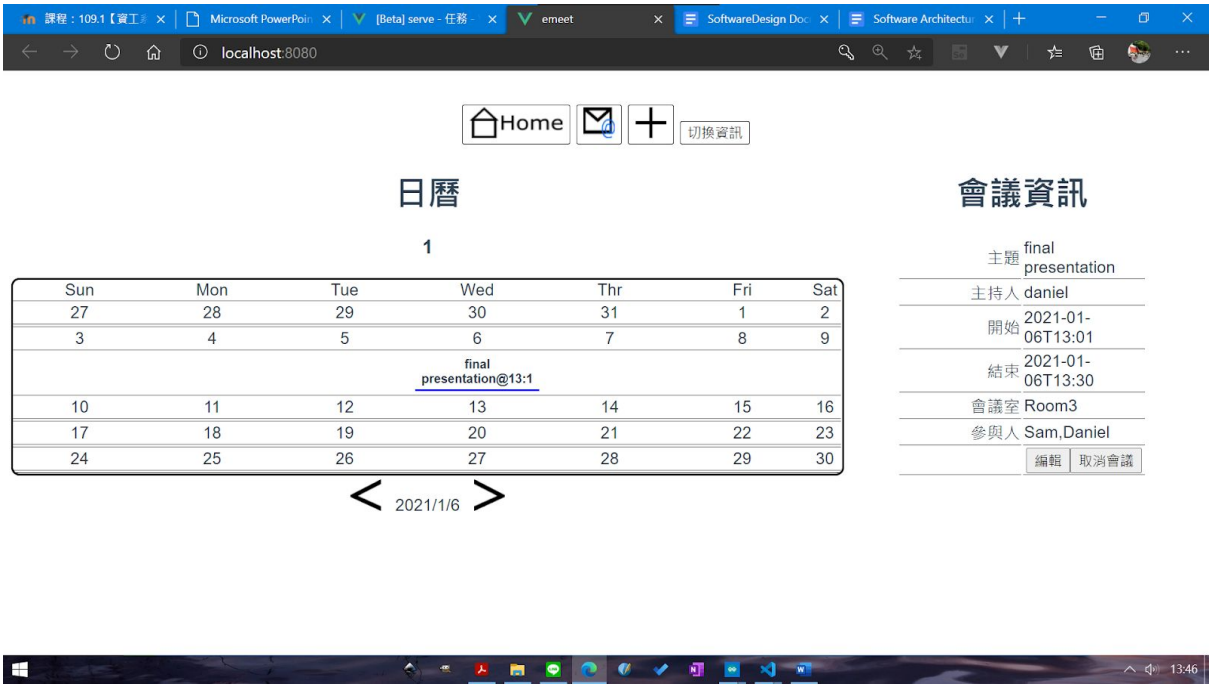
6.2.2 Signup Page



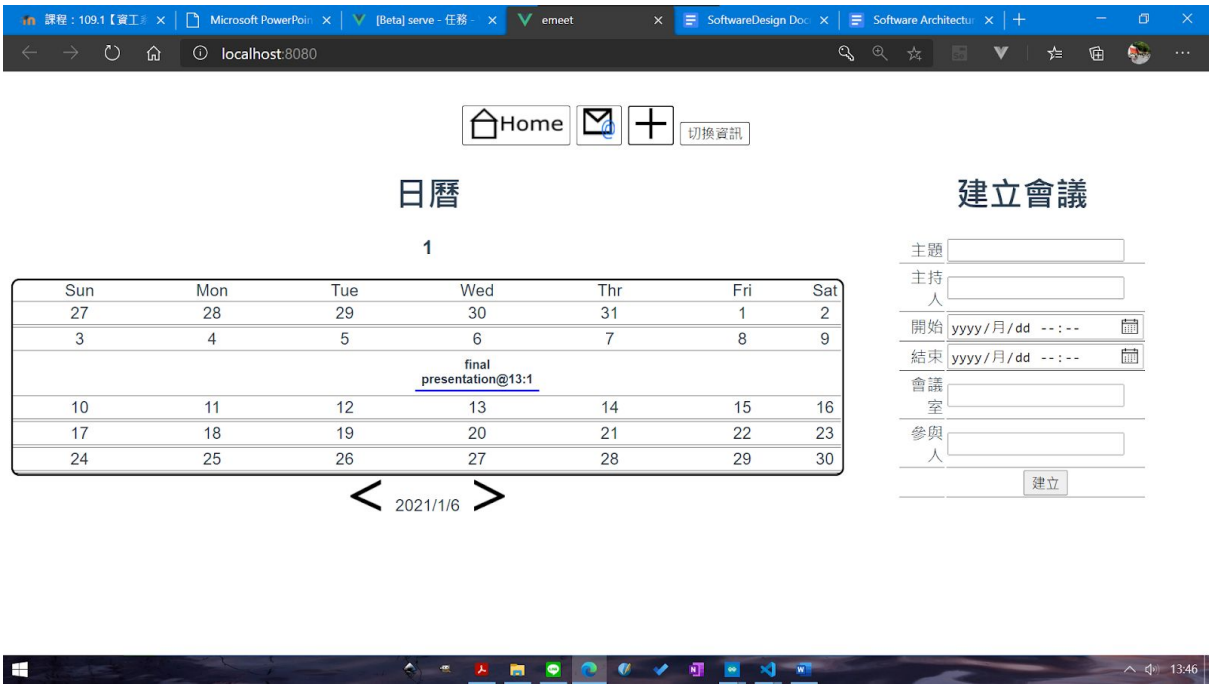
6.2.3 Reset Password



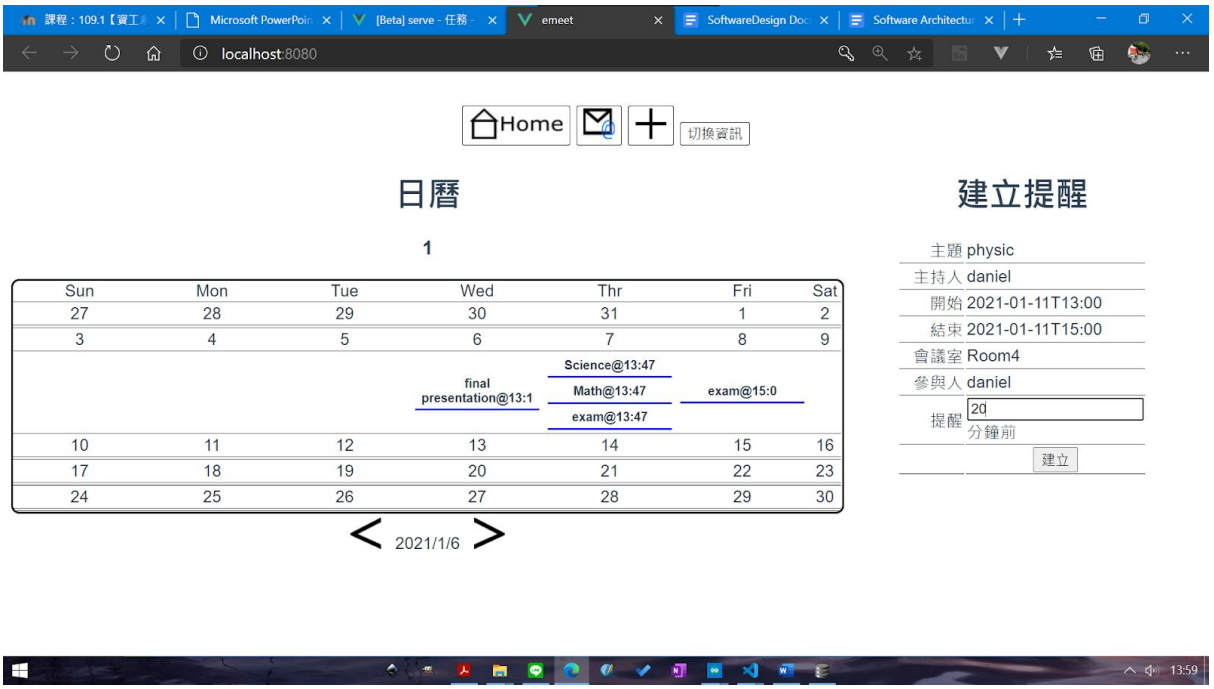
6.2.4 Main Page



6.2.6 Create Meeting Page



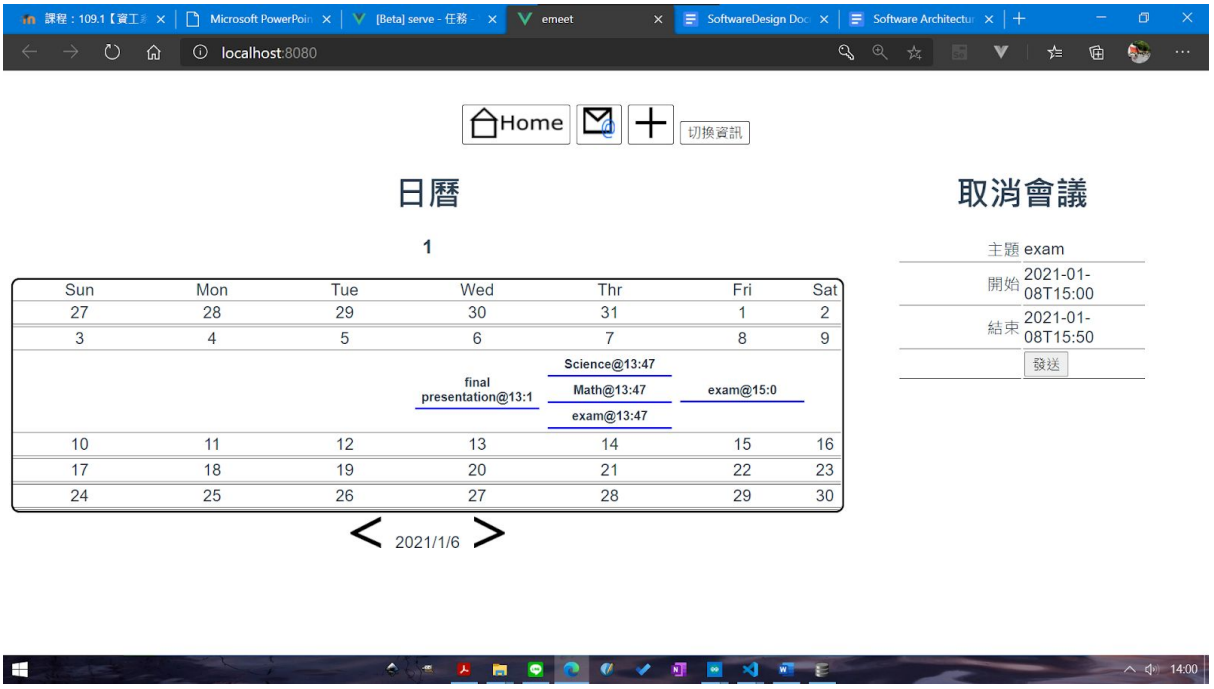
6.2.7 Create Reminder Page



6.2.8 Editing Meeting Page

The form used on create meeting was reused here for design simplicity.

6.2.9 Cancel Meeting Page



6.2.10 Meeting Room Information

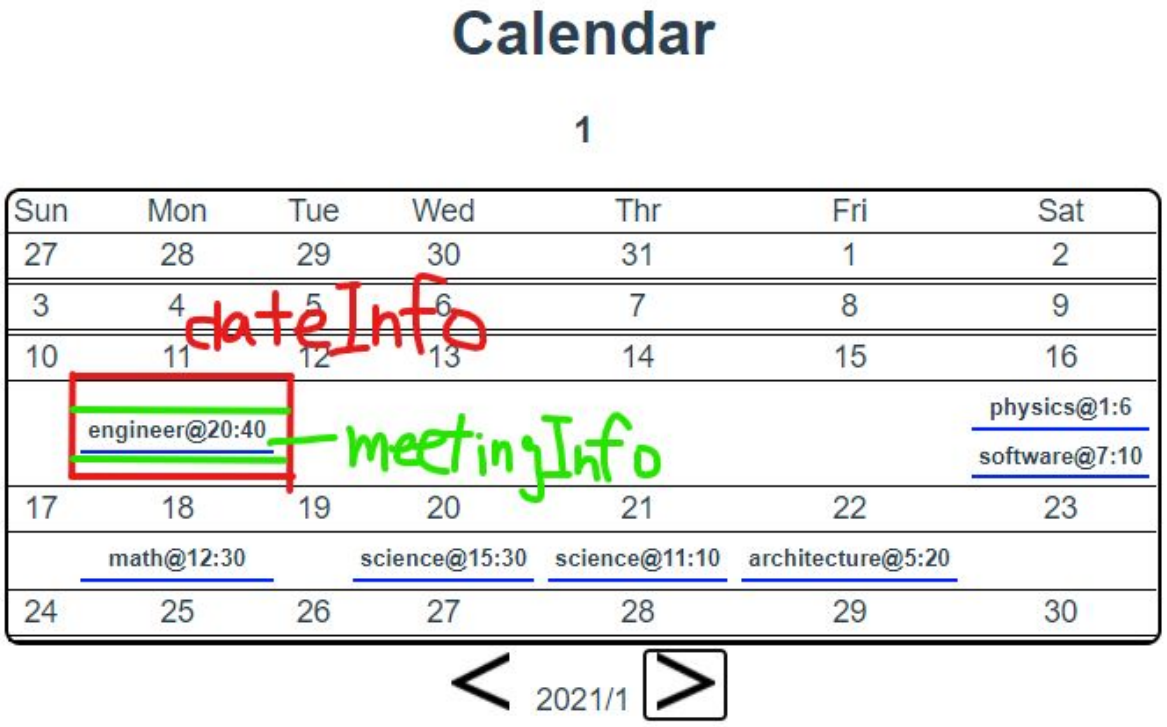
6.3 Screen Objects and Actions

6.3.1 Form

The most used UI component is form, which users enter some information in the specified type of input element, and the UI uses AJAX to transfer the content of form to backend server.

6.3.2 Calendar

This component could display all of the day in a month or a week depending on the usage and information contained in the calendar. A calendar is composed of three components: calendar, the root component, dateInfo, the component to hold the number of date and meeting events and meetingInfo, to hold the simple information of each meeting. Clicking the “>” and “<” buttons could switch the content of the calendar of last month and last month.



7.Requirement Matrix

This section reference SDD to SRS to show how software requirement are implement in architecture design

7.1 UI

The UI is simplified to only two different pages.

SRS	SDD
2.3.1 Signup	6.2.2 Signup
2.3.2 Login	6.2.1 Login
2.3.3 Home	6.2.3 Main
2.3.4 Create Meeting	6.2.5 Create Meeting
2.3.5 Create Google calendar reminder	6.2.6 Create Reminder
2.3.6 Meeting information	6.2.4 Meeting Selected
2.3.7 Cancel Meeting	6.2.8 Cancel Meeting
2.3.8 Room Information	6.2.9 Meeting Room Information

7.2 Program Logic

Most of the use cases are implemented through Web API, each API method relates to one use case requirement. The full information of API is in appendices. NaN means the use cases are not specified in the requirement document but added to API due to architectural design consideration.

API Index	Use Case
1. 新增User資料	2.1 Signup
2. User Login	2.2 Login
3. Get meeting	NaN
4. 新增Meeting資料	3.1 Create Meeting
5. 新增Google Reminder	3.3 Create Google Reminder
6. 編輯更改Meeting資料	NaN
7. 取消Meeting	3.2 Cancel Meeting
8. Get Room Info	NaN

8.Appendices

許世佑 et al.(2020), Room Reservation Assistant API Spec v1, National Taiwan University of Science and Technology, Software Engineering Course.