Software Design Document

For

Booking System

Version 1.0 approved

Prepared by B10815044 Hsieh, Jun-Yao
B10815054 Huang, Chen-En
B10815058 Pu, Chi-Hao
B10815057 Liao, Sheng-Hao

NTUST-SE-G8

目錄

1. Introduction 1
   - I. Purpose 1
   - II. Scope 1
   - III. Overview 1
   - IV. Definition, Acronyms and Abbreviations 1
   - V. References 2
2. System Overview 2
3. System Architecture 3
   - I. Architectural Design 3
   - II. Decomposition Description 3
   - III. Design Rationale 7
4. Data Design 7
   - I. Data Description 7
   - II. Data Dictionary 8
5. Component Design 9
   - I. Book System 9
   - II. DataBase API 10
   - III. User Interface 12
   - IV. Google Calendar API 14
6. Human Interface Design 15
   - V. Overview of User Interface 15
   - VI. Screen Images 15
   - VII. Screen Object and Actions 19
7. Requirement Matrix 25
8. Appendices 27
   - I. Setup and Configuration 27
   - II. Tools 27
   - III. Environment 28
   - IV. Contribution of Team members 28

Revision History

Name	Data	Reason for Change	Version
Draft	2021/12/29	First version	1.0

1. Introduction

I. Purpose

The Software Design Document is a document that is supposed to provide documentation which will be used to focus on software development by providing the details for how the software should be built. Within the Software Design Document are narrative and graphical documentation of the software design for the project including use case models, sequence diagrams, collaboration models, object behavior models, and other supporting requirement information.

II. Scope

This document is intended to give a detailed technical description of the Book System software project. It specifies the structure and design of some of the modules discussed in the SRS. It also displays some of the use cases that had transformed into sequential and activity diagrams. The class diagrams show how the programming team would implement the specific module.

III. Overview

This document is written according to the "SDDTemplate".
Sections 3 – 5 contain discussions of the designs for the project with diagrams, section 6 shows samples of UI from the system, and section 7 contains the class diagrams.

The appendices contain the setup and configuration needed for the Book System, a list of tools and environment used in the entire project, along with the time contribution of team members.

IV. Definition, Acronyms and Abbreviations

Term	Definition
API	Abbreviation of Application Interfaces

UI	Abbreviation of User Interface
Database	It’s a system that is used to save data
MySQL	One of SQL databases
User	Normal user can raise or modify a meeting, but can’t create meeting rooms
Manager	Manager is one of users, but the manager can create or delete meeting room
Room	Corresponds to the space where the entity exists, so only one event can exist at the same time
Event	Corresponding to a period of time, may be meetings, teaching activities, etc..
Attendee	Participants of the meeting, save email address to send notification
FRP	Fast Reverse Proxy

V. References

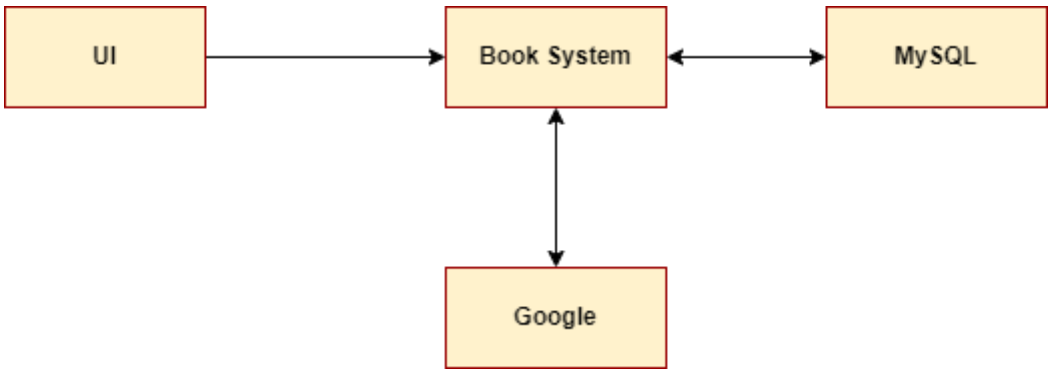
- example_design_SDD
- SDDTemplate
- Software Design Document
- Software Design Document, Testing, and Deployment and Configuration Management

2. System Overview

Give a general description of the functionality, context and design of the Book System. In order to make the system as flexible as possible, the system is divided into 4 parts: Main system, UI, Database, Google calendar. So that we can explicitly know which part is responsible for who, and also it’s easier to debug.

3. System Architecture

I. Architectural Design



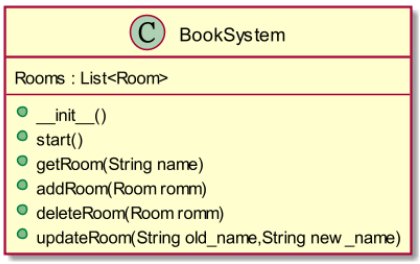
II. Decomposition Description

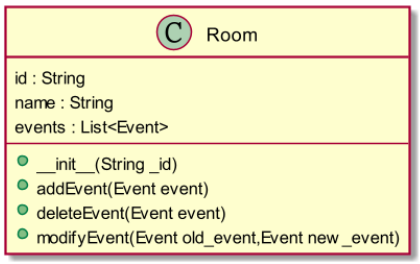
Take modify event function to make a example:

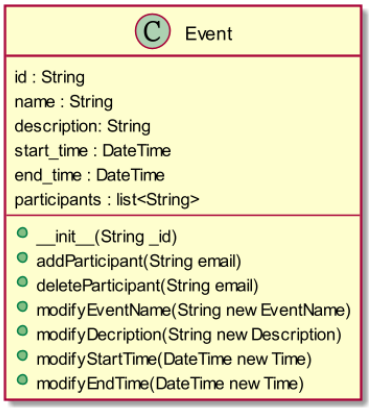
User trigger the modify event function, the BooksystemUI send request to BookInterface, then BookInterface received the request, contact BookSystem to modify, Booksystem find the meeting room and designated event, and the Event class modity event, return the new event to Booksystem and return to BookInterface after modification, and the BooksystemUI display the new event.

Object diagrams

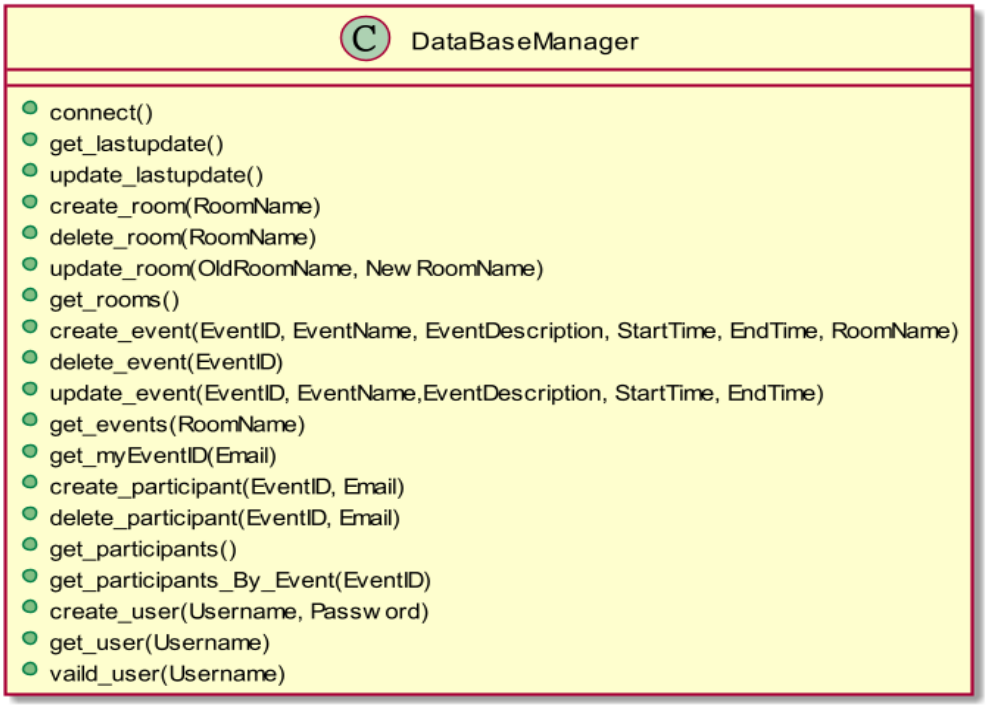
- **BookSystem**



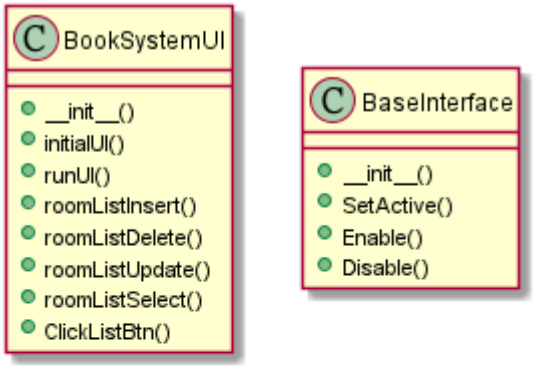


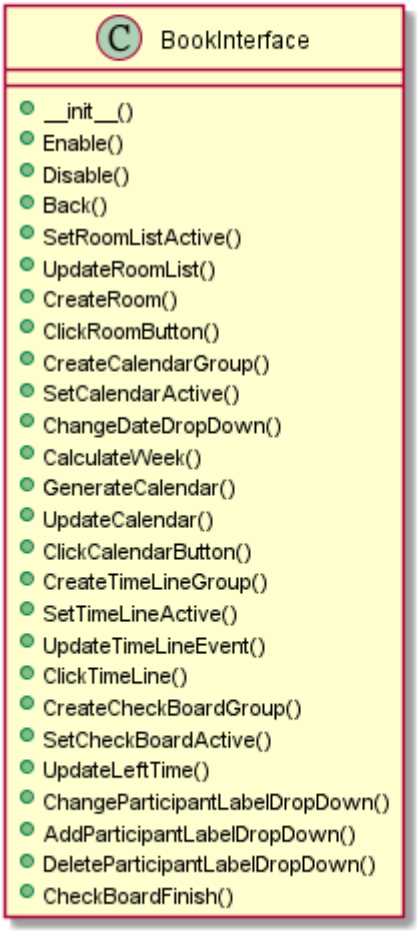


- **DataBase**

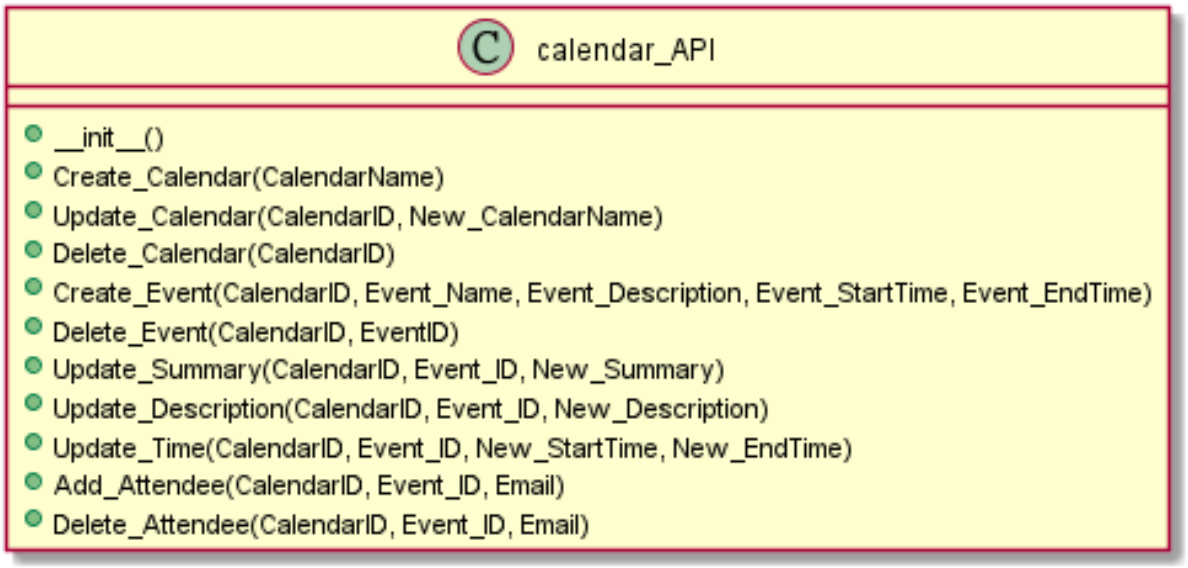


- **UI**

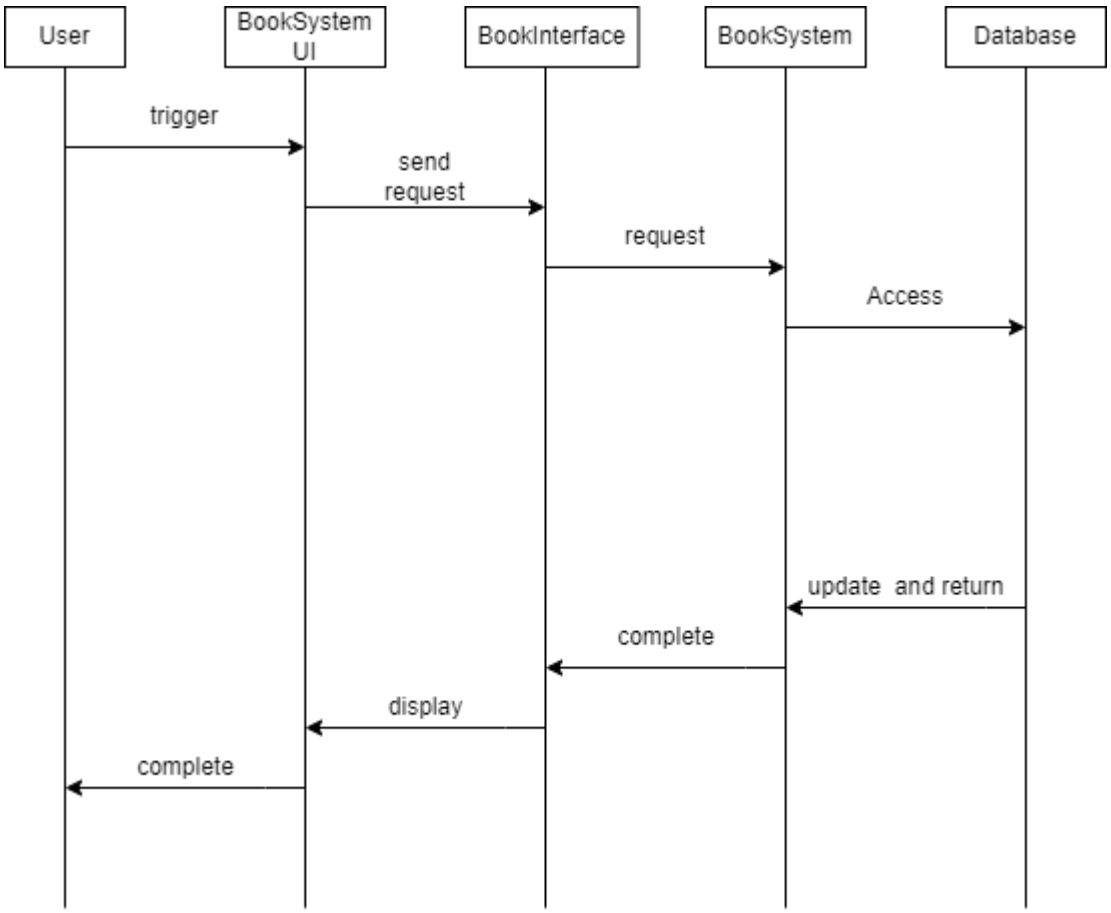

BookSystemUI
__init__()
initialUI()
runUI()
roomListInsert()
roomListDelete()
roomListUpdate()
roomListSelect()
ClickListBtn()
BaseInterface
__init__()
SetActive()
Enable()
Disable()


BookInterface
__init__()
Enable()
Disable()
Back()
SetRoomListActive()
UpdateRoomList()
CreateRoom()
ClickRoomButton()
CreateCalendarGroup()
SetCalendarActive()
ChangeDateDropDown()
CalculateWeek()
GenerateCalendar()
UpdateCalendar()
ClickCalendarButton()
CreateTimeLineGroup()
SetTimeLineActive()
UpdateTimeLineEvent()
ClickTimeLine()
CreateCheckBoardGroup()
SetCheckBoardActive()
UpdateLeftTime()
ChangeParticipantLabelDropDown()
AddParticipantLabelDropDown()
DeleteParticipantLabelDropDown()
CheckBoardFinish()

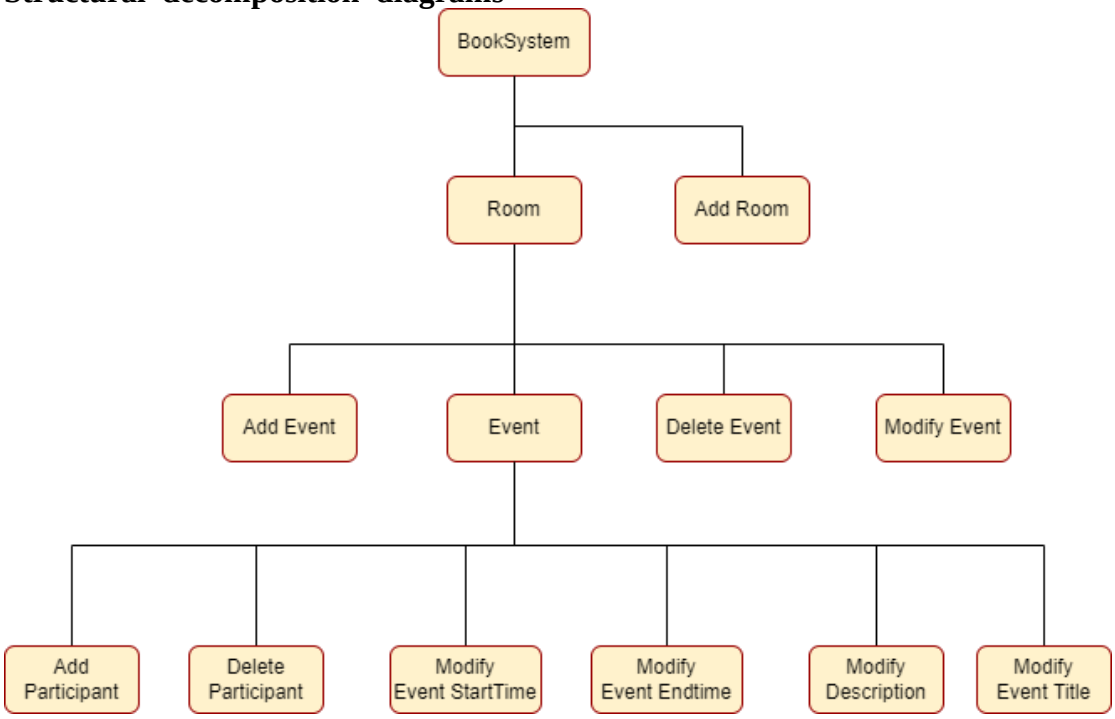
- **Google Calendar API**



Sequence diagram



Structural decomposition diagrams



III. Design Rationale

We use BookInterface class to be the core of the system, it connect most the class in system because it receive request from user operating UI, and send request to the class in application package, the class in application package will complete the request and send back to BookInterface class, then BookInterface send the consequence to BookSystemUI.

We make one class to be the core of the system instead of a separate system, because there are many functional requirements in the system, so it needs to have one central class to contact the other functional class.

4. Data Design

I. Data Description

The system uses MySQL as database and MySQL Connector for python to communicate with the database.

- **Rooms**: is used to store meeting rooms, RoomID is a serial number of Google

Calendar's calendar, RoomName is the meeting room's name that shows in the UI.

- **Events**: is used to store meeting's information, RoomID is a serial number of Google Calendar's event, RoomName is used to search, others are regular records for the meeting.
- **Participants:** is used to store participant's information, EventID is for searching, Email is user's identification.
- **Users: i**s used to store user's information.
- **Synchronize:** is used to Synchronize Main system and Database, the LastUpdate is used to check the last change's time.

II. Data Dictionary

	Field	**Type**	**Null**	**Default**
Rooms	RoomID	varchar(60)	No	
	RoomName	varchar(20)	No	
Events	EventID	varchar(30)	No	
	EventName	varchar(20)	No	
	EventDescription	varchar(400)	Yes	NULL
	StartTime	datetime	No	
	EndTime	datetime	No	
	RoomName	varchar(20)	No	
Participants	EventID	varchar(30)	No	
	Email	varchar(40)	No	
Users	UserName	varchar(40)		
	Password	varchar(100)		

Synchronize	LastUpdate	datetime	No	CurrentTimeStamp

5. Component Design

I. Book System

BookSystem

Function	**Description**
__init__	Construct all needed member
start	Start the booksystem
update	Update all data from database
check_db_update	Check if need update then act
getRoom(name)	Return room by room's name
getRoomById(id)	Return room by room's id (google calendar generate)
getRoomEvents(room_name)	Return all event from room by room's name
addRoom(room)	Add a room to system
deleteRoom(room)	Delete a room from system
updateRoom(old_name,new_name)	Update a room from system
addParticipant(email)	Add participant to global participant pool
getUserEvents(CurrentUser)	Get an user's all event from database

garbage_event_collection	Delete past events

Room

Function	**Description**
addEvent(event)	add an event to room
deleteEvent(event)	delete an event from room
updateEvent(new_event)	update an event from room
getEvent(name)	return event by event's name
getEventById(id)	return event by event's id
getEventParticipant(event_id)	get all participant from an event
exist()	check if this room exist in database

Event

Function	**Description**
update_participants(_participants)	update participants to new participants
update(new_event)	update event info to new event
in_event(email)	check if email is one of participants
exist()	check if this event exist in database

II. DataBase API

Function	**Description**
connect()	Initialize and connect to Database
get_lastupdate()	Get system's last update time
update_lastupdate()	If data modify will call to update the last update time

create_room(RoomName)	Add a room record to the database
delete_room(RoomName)	Delete Room from the database
update_room(OldRoomName, NewRoomName)	Update RoomName from OldRoomName to NewRoomName
get_rooms()	Search and return all of rooms
create_event(EventID, EventName, EventDescription, StartTime, EndTime, RoomName)	Add a Event record to the database
delete_event(EventID)	Delete event from the database
update_event(EventID, EventName,EventDescription, StartTime, EndTime)	Update Event's information with new values
get_events(RoomName)	Get all of rooms from specific room
get_myEventID(Email)	Get all of event's by Email
create_participant(EventID, Email)	Add a participant to specific event
delete_participant(EventID, Email)	Delete a participant from specific event
get_participants()	Get all of participants from all of events
get_participants_By_Event(EventID)	Get all of participants from specific event
create_user(Username, Password)	Register an user to the Book System
get_user(Username)	Check whether user has already registered or not
vaild_user(Username)	Return user's password to match user's input

III. User Interface

BookSystemUI

__init__(BookSystem)	Construct the UI
initial()	Construct GUI window and menu
runUI()	Start GUI main loop
roomListInsert(name)	Insert a new room name to ListBox
roomListDelete(name)	Delete a new room name from ListBox
roomListUpdate()	Update a new room name from ListBox
roomListSelect(index)	Select an item from ListBox
ClickListBtn(_btnNum)	Switch UI canvas

BaseInterface

__init__(_parent)	Construct the canvas
SetActive(_value)	Control interface display or off
Enable()	Show and setting the interface
Disable()	Hide the interface

LogInInterface(BaseInterface)

__init__(_parent,_bookSystem)	Construct the canvas
Enable()	Show and setting the interface
Disable()	Hide the interface
LogInAccount()	Login
SignAccount()	Sign an account

BookInterface(BaseInterface)

__init__(_parent,_bookSystem)	Construct the canvas
Enable()	Show and setting the interface
Disable()	Hide the interface
Back()	Back to previous UI
UpdateRoomList()	Update the room list
CreateRoom()	Create the room button on room list
ClickRoomButton(_roomName)	Choose a meeting room and enter the calendar UI
ChangeDateDropDown(_mode)	Choose year and month
CalculateWeek(year,month)	Calculate the week of the target date
GenerateCalendar()	Create the calendar
UpdateCalendar()	Update the calendar
ClickCalendarButton(_day)	Select the date and enter the timeline UI
UpdateTimeLineEvent()	Update the event on timeline
ClickTimeLine()	Select the meeting start time and enter the booking interface
UpdateLeftTime()	Update end list of end time
CheckFormat()	Check the format of the data entered by the user
CheckBoardFinish()	Book complete

UsersBookInterface(BaseInterface)

__init__(_parent,_bookSystem)	Construct the canvas
Enable()	Show and setting the interface
UpdateEventList()	Update the event list
CreateEvent()	Create event label on event list

IV. Google Calendar API

Function	Description
__init__()	Construct Google Calendar API service
Create_Calendar(CalendarName)	Create a calendar
Update_Calendar(CalendarID, New_CalendarName)	Update name of calendar
Delete_Calendar(CalendarID)	Delete calendar
Create_Event(CalendarID, Event_Name, Event_Description, Event_StartTime, Event_EndTime)	Create an event in calendar
Delete_Event(CalendarID, EventID)	Delete event
Update_Summary(CalendarID, Event_ID, New_Summary)	Update the summary of event
Update_Description(CalendarID, Event_ID, New_Description)	Update the description of event
Update_Time(CalendarID, Event_ID, New_StartTime, New_EndTime)	Update the time of event
Add_Attendee(CalendarID, Event_ID, Email)	Add attendee into event
Delete_Attendee(CalendarID, Event_ID, Email)	Delete attendee from event

6. Human Interface Design

V. Overview of User Interface

The system interface will allow users to make reservations, browse all meeting rooms and ongoing events, and book meeting rooms.

The first screen after logging in is the reservation system, which displays a list of meeting rooms, and the reservation process uses a step-by-step approach that allows users to reserve a meeting room correctly.

Users can switch to the appointment interface or view their own events through the list on the left. If the user's login account is an administrator account, there will be an additional administrator option in the list, allowing the administrator to add or remove meeting rooms.

VI. Screen Images

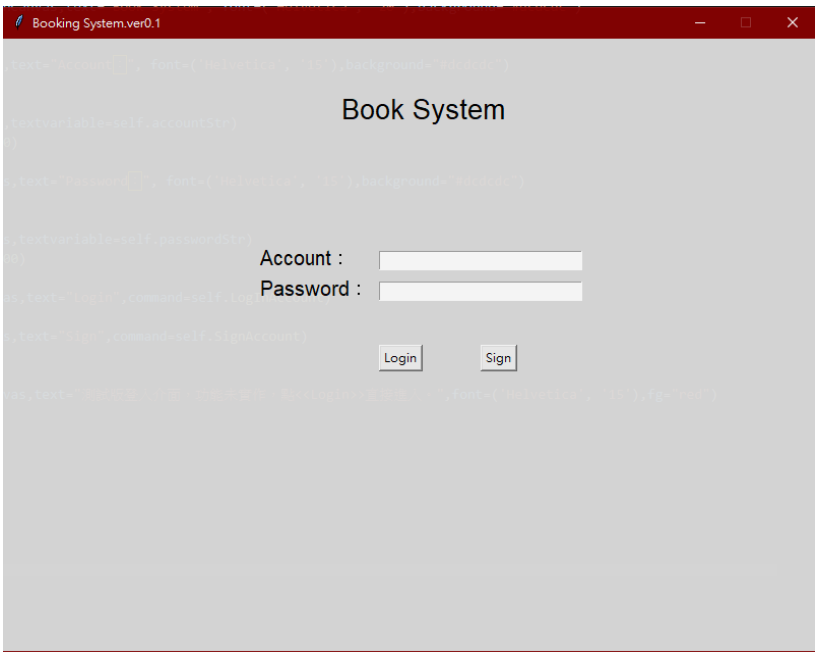


Figure 1.Login UI

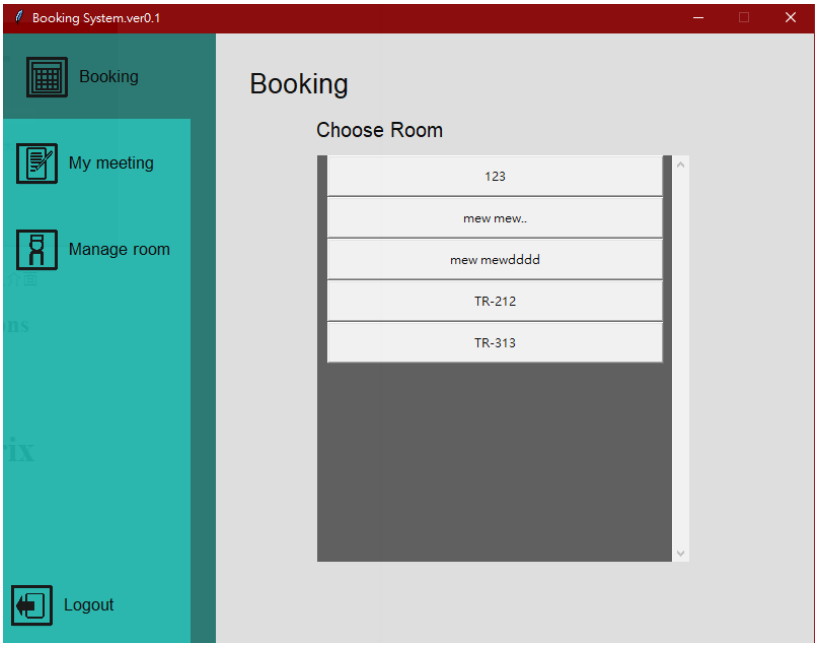


Figure 2. Room List

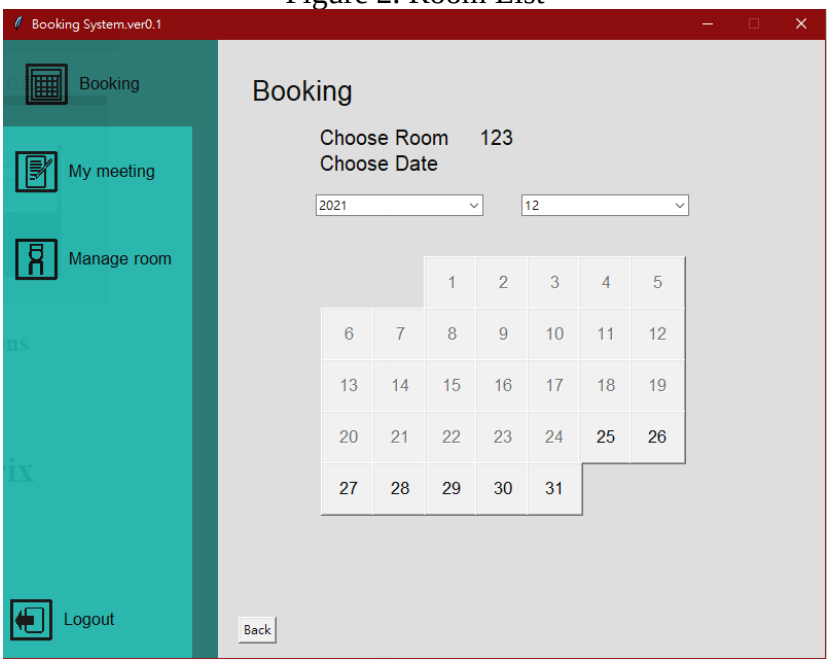


Figure 3. Calendar

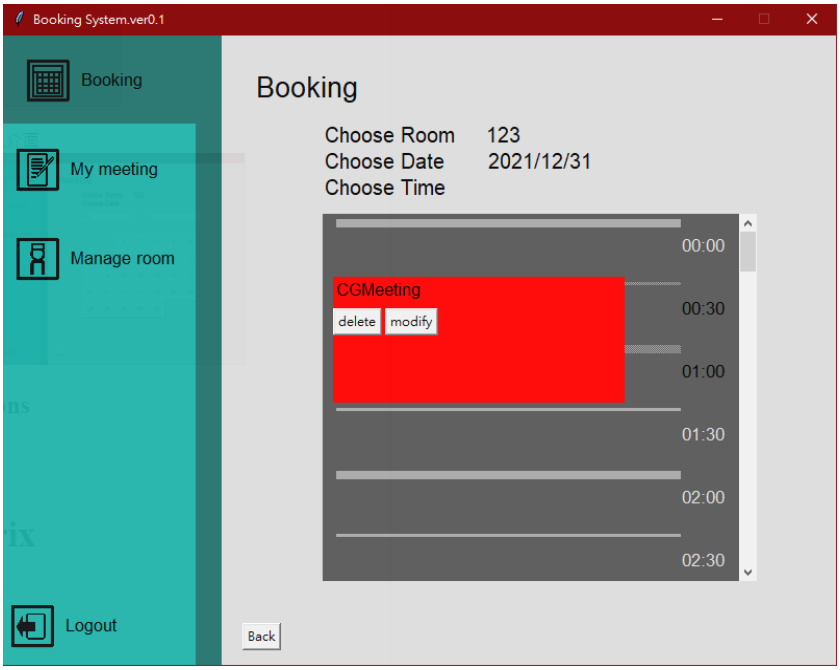


Figure 4. Timeline

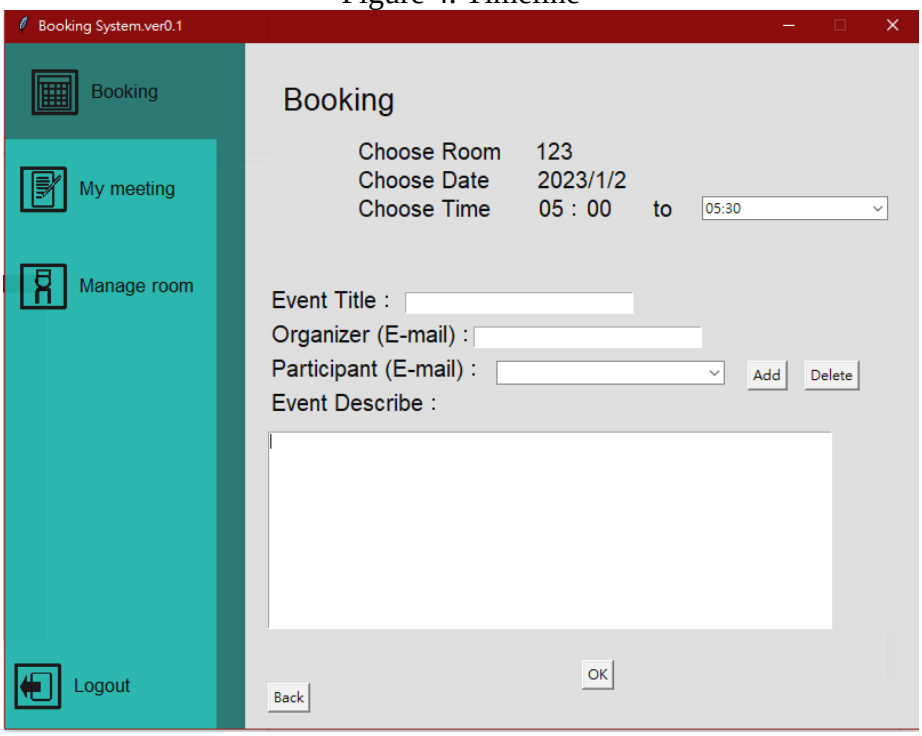


Figure 5. Booking

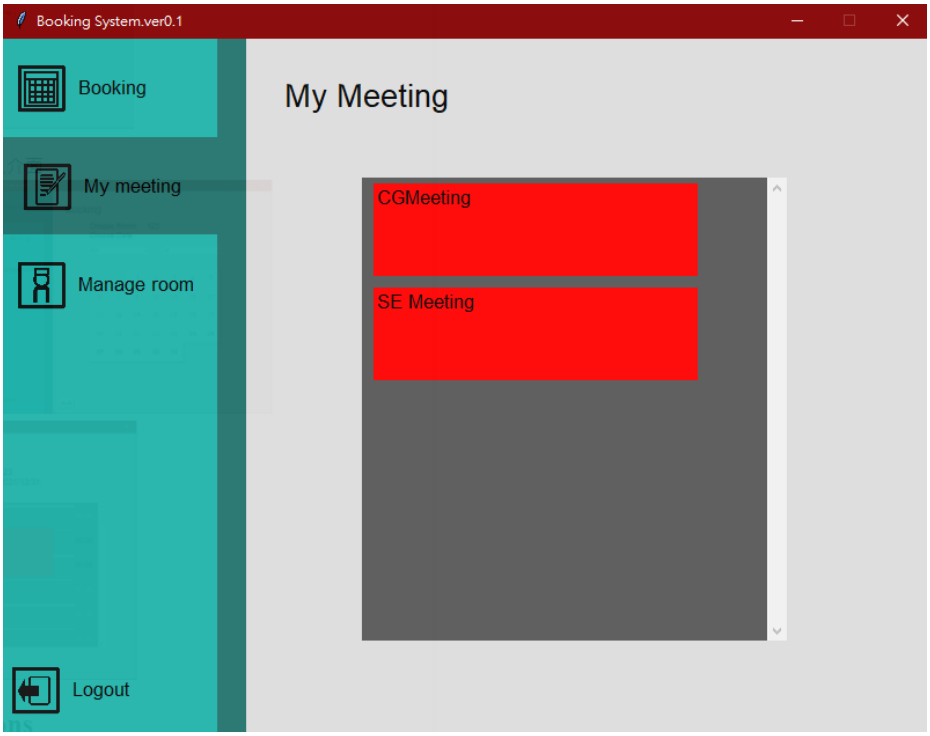


Figure 6. My Meeting

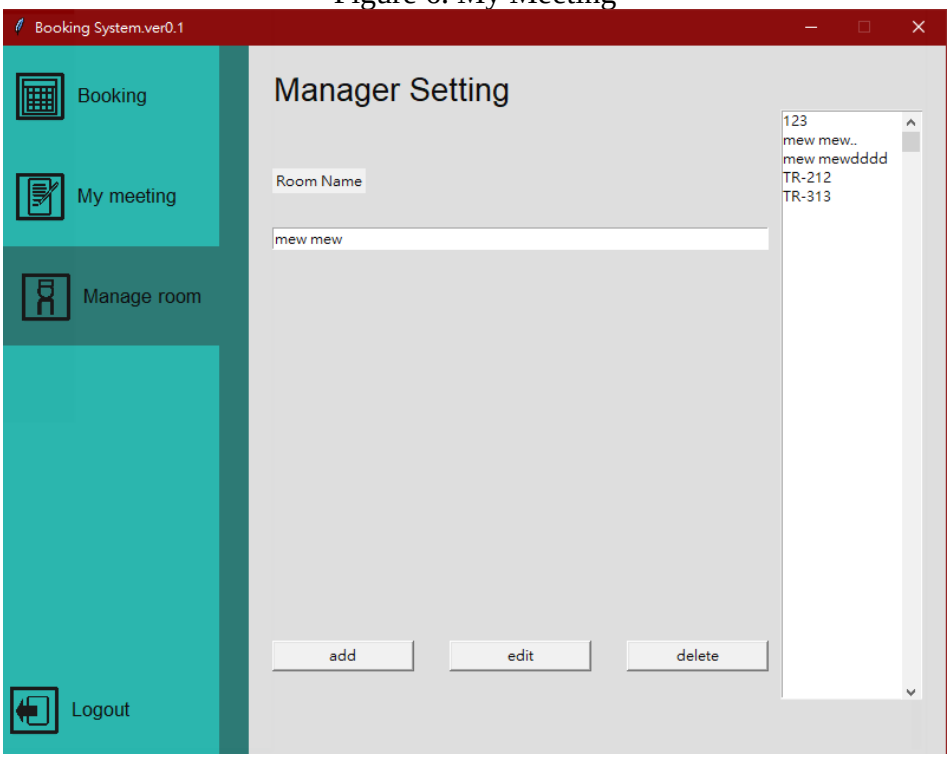
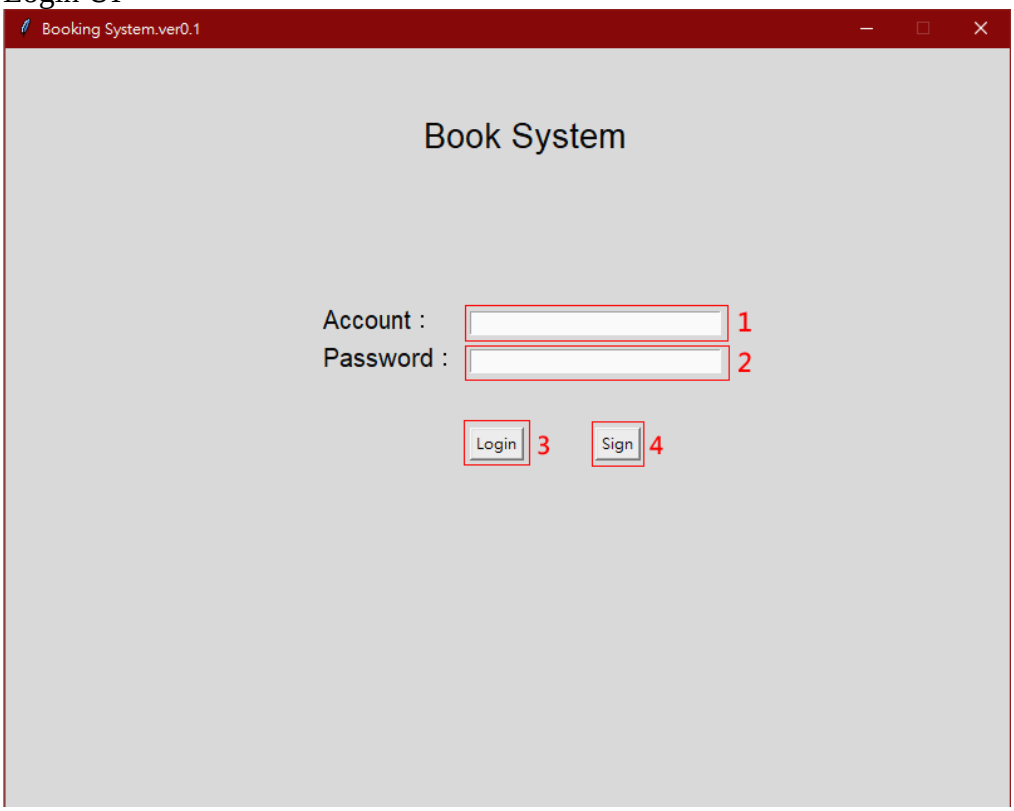


Figure 7. Manager Options

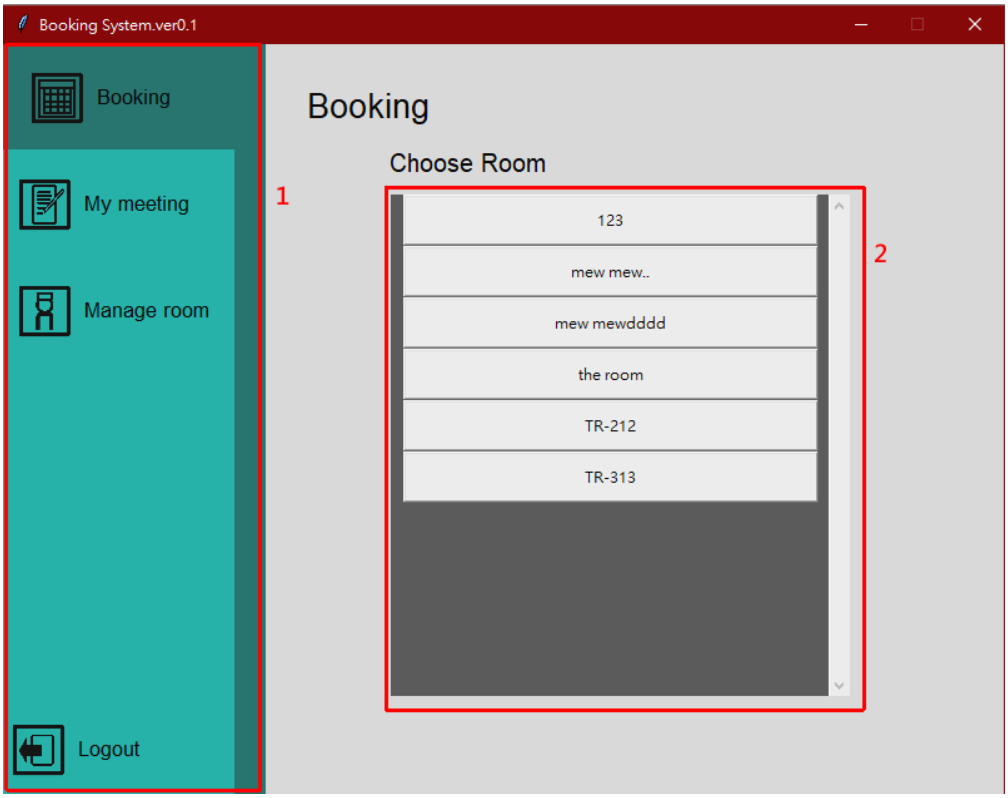
VII. Screen Object and Actions

1. Login UI



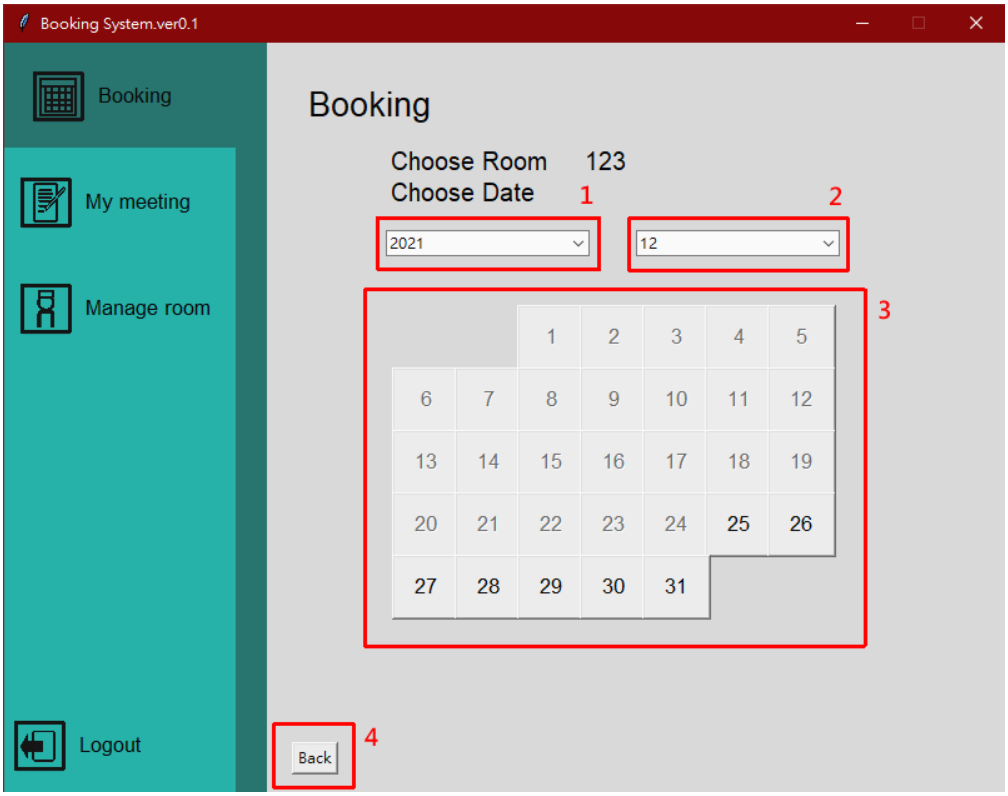
1.Account	Input Email account to register
2.Password	Input password from the account
3.Login Button	Click to login the system
4.Register Button	Click to register account the system

2. Initial UI



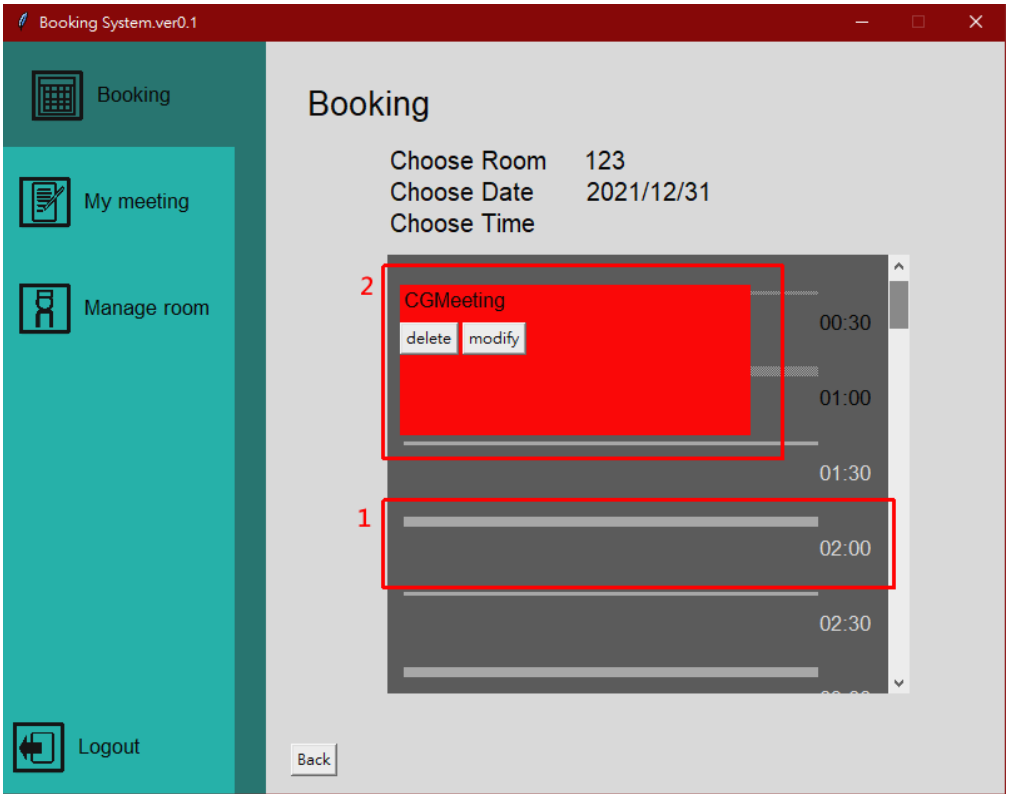
1. Function list	Through this list, you can switch to other functional interfaces
2. Meeting rooms list	List all available meeting rooms and click on them to enter the reservation process.

3. 日歷



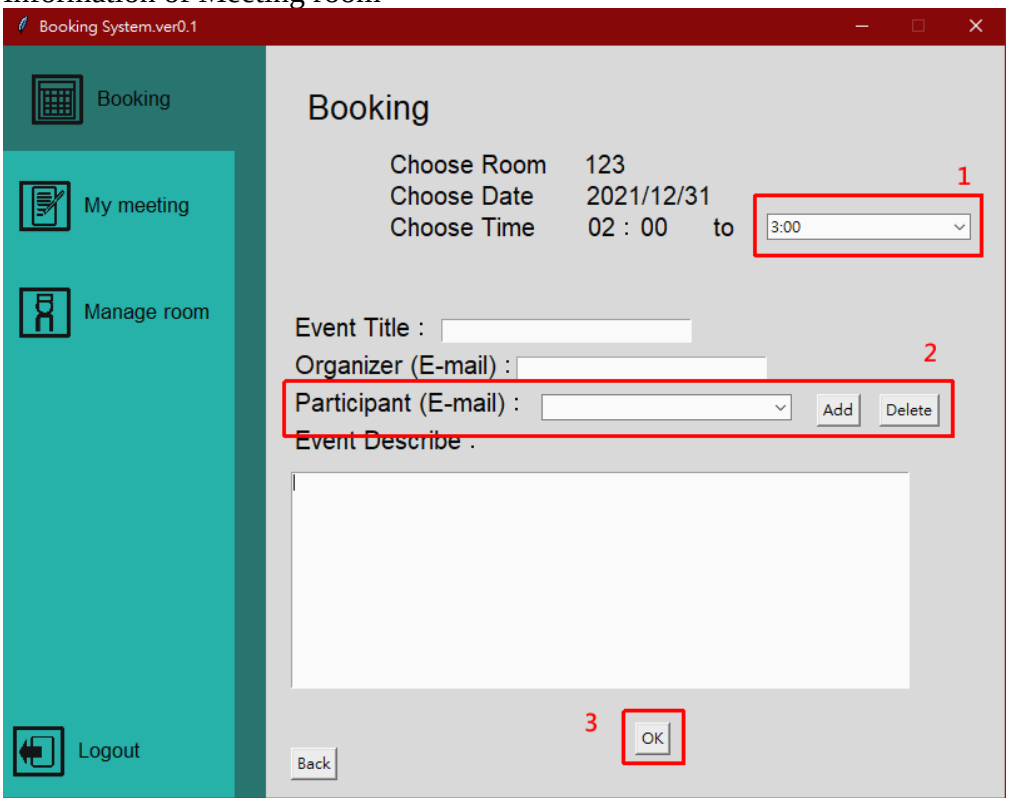
1. Dropdown(year)	Click to select the year you want to reserve
2. Dropdown(Month)	Click to select the month you want to reserve
3. Calendar	Click to select the date you want to reserve
4. Back button	Back to previous UI

4. Events Timeline



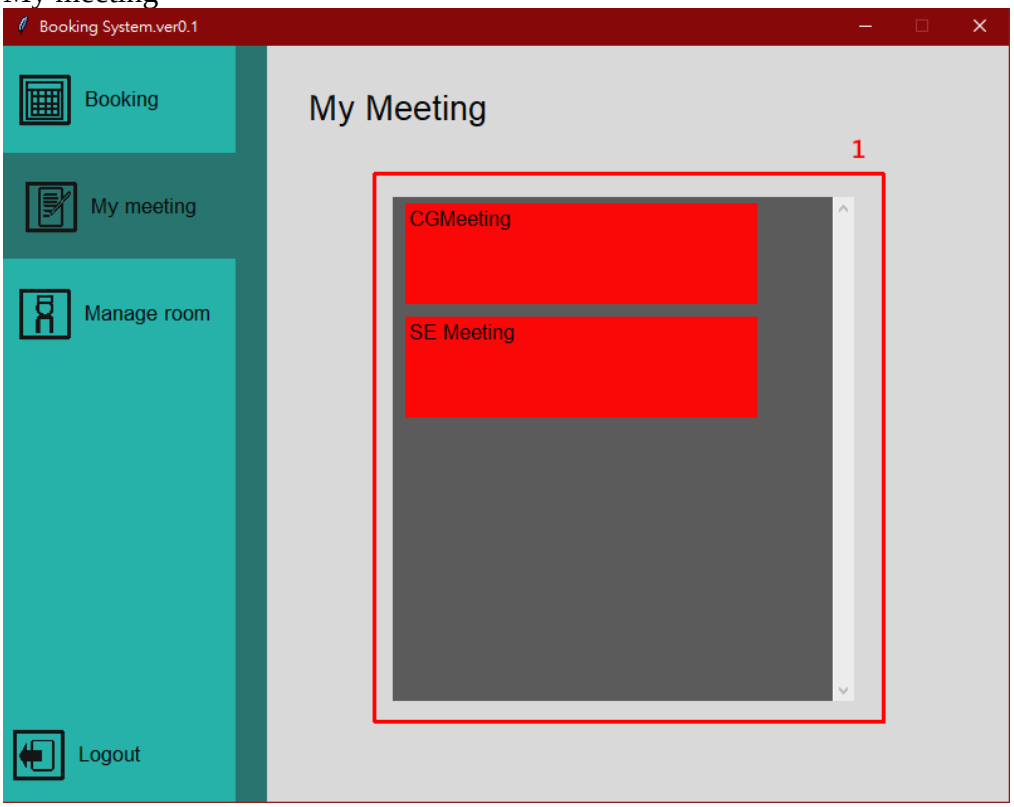
1. Timeline	Click to select the meeting start time
2. EventStamp	Timelines that have already been booked will be covered by this red mark and can be modified or deleted if the owner of the event is the owner.

5. Information of Meeting room



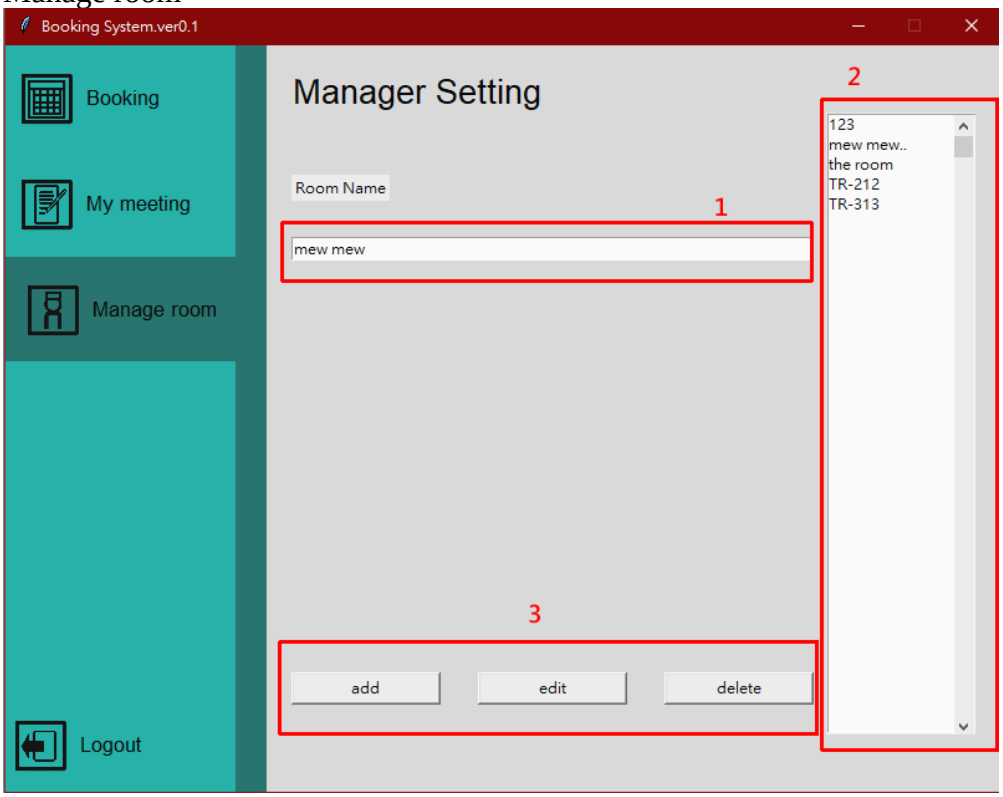
1. Dropdown(End time)	Click to select the end time of the meeting
2. Participant	After entering the e-mail address of the participant in the left column, you can add the participant through the Add button, or click the drop-down list to select an existing participant and press Delete to remove it.
3. OK button	Click to confirm change

6. My meeting



1. Event list	List events that the user owns or participates in

7. Manage room



1. Room Name	Enter the name of the room you want to add or modify.
2. Room list	Show all of rooms
3. Change Options	Click on the three buttons below to add, modify the name, and remove actions respectively

7.Requirement Matrix

ID	functional requirement name	Functional requirement description	Corresponding Function that implemented in system
1	Create meeting room	Create a meeting room in database	BookSystem.addroom(Room room) in application package
2	Create Meeting	create a upcoming meeting in the meeting	Room.addEvent(Event event)

		room	in application package
3	Update Meeting Title	modify the title of the meeting that already existed in meeting room	Room.modifyEvent(Event old_event,Event new_event) Event.modifyEventName(String new EventName) in application package
4	Update Meeting Description	modify the description of the meeting that already existed in meeting room	Room.modifyEvent(Event old_event,Event new_event) Event.modifyDescription(String new Description) in application package
5	Update Meeting Time	modify the time of the meeting that already existed in meeting room	Room.modifyEvent(Event old_event,Event new_event) Event.modifyStartTime(DateTime new Time) Event.modifyEndTime(DateTime new Time) in application package
6	Add Attendee	Add participant into the meeting that already existed in meeting room	Room.modifyEvent(Event old_event,Event new_event) Event.addParticipant(String email) in application package
7	Remove Attendee	Remove the participant from the meeting that already existed in meeting room	Room.modifyEvent(Event old_event,Event new_event) Event.deleteParticipant(String email) in application package
8	Delete Meeting	Remove the meeting	Room.deleteEvent(Event

		that already existed in meeting room	event) in application package
9	Delete Meeting Room	Delete the Meeting Room	Booksystem.deleteRoom(Room room) in application package
10	Update meeting room name	modify the name of the meeting room that already existed	BookSystem.updateRoom(String old_name,String new_name) in application package

8. Appendices

I. Setup and Configuration

I.1 MySQL

I.1.1 Create a database for the system
I.1.2 Add Manager to manage the Database
I.1.3 Create Tables to store data
I.1.4 Create some rules for data management

I.2 FRP

I.2.1 Because we don't static IP address for server, so we need to access the service through FRP
I.2.2 Need to find a server to activate the offer the service
I.2.3 Use a local computer as DB server

II. Tools

- Git
- GitHub Desktop :Not Required, but recommended
- Visual Studio Code: Not Required, but recommended
- Google Calendar API
- MySQL

III. Environment

- Windows 10 or above
- Python 3.7

IV. Contribution of Team members

Hsieh, Jun-Yao **(Project Manager)** **(Developer)**	1. Meeting Host 2. Documents Management 3. Develop and Manage Database 4. Help to develop the main system 5. System Test 6. Job assignment and coordinate
Huang, Chen-En **(Developer)**	1. Google Calendar API interface 2. Functional requirement implement 3. Technical writer
Pu, Chi-Hao **(Art)** **(Developer)**	1. GUI design 2. Technical writer 3. System tester
Liao, Sheng-Hao **(Program Manager)** **(Developer)**	1. Core component design 2. Core component implement 3. Run & Build manual writer 4. Google account provider 5. Technical writer 6. System tester