Software Requirements Specification

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

Revision History

Name	Data	Reason for Change	Version
Draft	2021/11/09	First version	1.0

1. Introduction

1. Purpose

The document is mainly about the Book System's details, such as how it works, operates, and resolves problems. The Book System is suitable for companies to arrange their meeting rooms. Companies can raise meetings or check available meeting rooms through the system, the document will talk about in more detail below.

2. Glossary

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

3. Intended Audience and Reading Suggestions

In order to avoid these tragedies, a good room management system is necessary, and the system must be real-time, convenient and stable. The booking system that we have developed can perfectly manage the rooms, events, and personnel to ensure that they do not conflict with each other, and after using this system, you can focus on the content of the event instead of spending a lot of time arranging the venue and adjusting the time.

After reading the first chapter, you will understand the objective and usage situation of the system, after reading the second chapter, you will understand the operation flow and principle of the system, after reading the third chapter, you will understand how the underlying system works, after reading the fourth chapter, you will understand that the system not only has basic functions, but also contains the design made by considering many aspects.

4. Product Scope

This system provides a solution for event or personnel conflict, using information technology to integrate all data together, determine conflict or not, and allow users to quickly know which meetings they have, send reminder emails, and reduce the chance of meeting personnel absence.

5. References

Software Engineering 10th Edition by Ian Sommerville
For more information about software engineering, please check the textbook.

2. Overall Description

1. System Environment

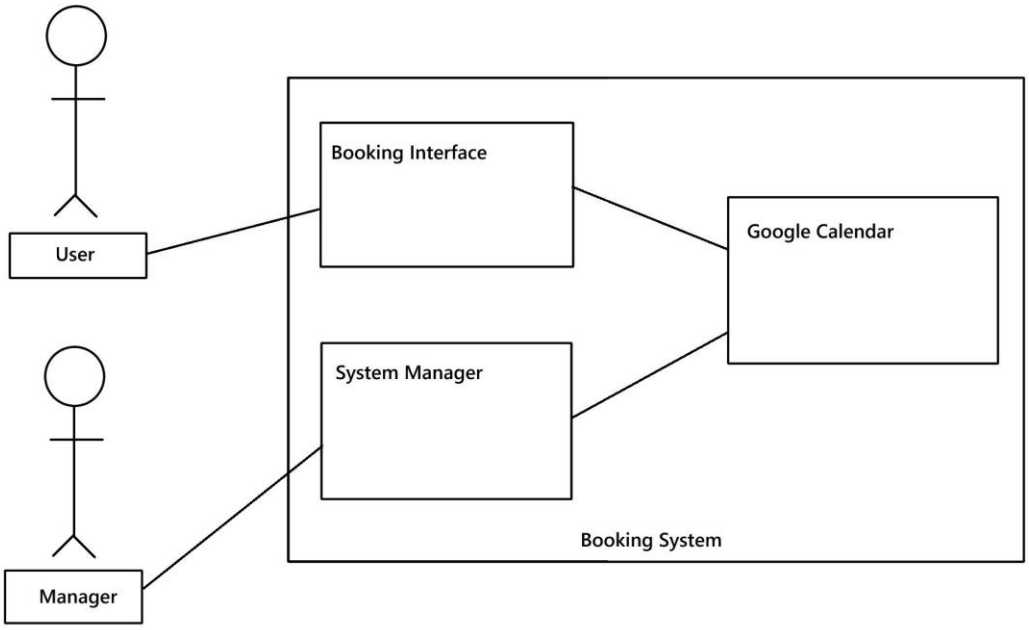


Figure 1 - System Environment

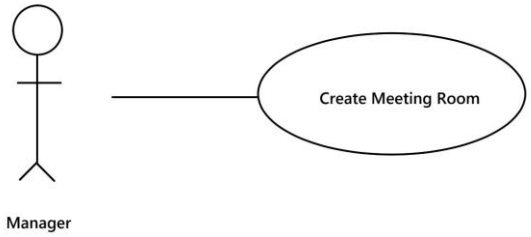
The Booking System will have two different types of users, one is normal user who can book meeting room, create meeting, check meeting they participant. Another is The System Manager who can manage all of the meeting rooms.

2. Functional Requirements Definition

The part descripts the using cases of The Book System, normal users are the main participant.

2.2.1 Manager Use Case

Use case: **Create Meeting Room**
Diagram:



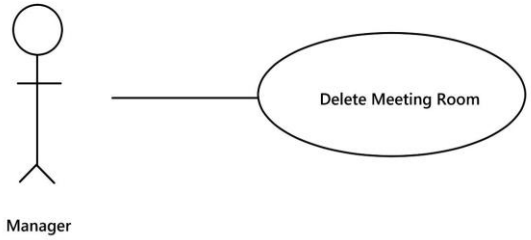
Brief Description
System manager adds the existing room into the meeting room list

Initial Step-By-Step Description
Before this use case can be initiated, the Manager has already accessed the main page of the System Manager.

1. System manager enters the name of the meeting room
2. Press the add button
3. System will add the meeting room into the database and meeting room list

Xerf: Section 3.2.1, Create Meeting Room

Use case: **Delete Meeting Room**
Diagram:



Brief Description
System manager deletes the existing room in the meeting room list

Initial Step-By-Step Description
Before this use case can be initiated, the Manager has already accessed the main page of the System Manager.

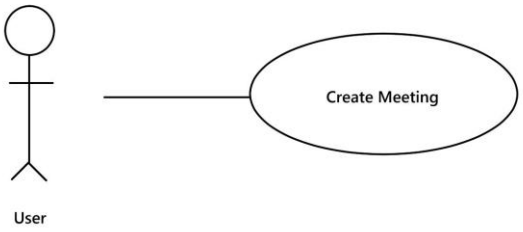
1. Select the meeting that want to be deleted from the list
2. Press the delete button

3. System will delete the meeting room from the database and meeting room list

Xerf: Section 3.2.9, Create Meeting Room

2.2.2 User Use Case

Use case: **Create Meeting**
Diagram:



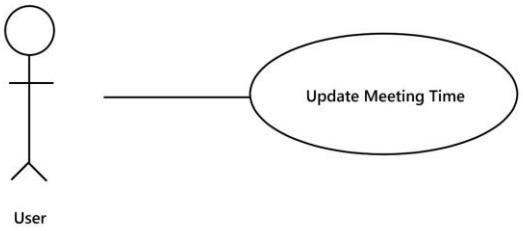
Brief Description
User raises a meeting in the meeting room

Initial Step-By-Step Description
Before this use case can be initiated, the User has already accessed the main page of the Booking Interface.

1. Select the meeting that want to be book from the list
2. UI will show the calendar of the meeting room
3. Select the date from the calendar
4. UI will show the schedule of the date
5. Select unbook date
6. User can raise events there
7. Enter event's info, such as title, hoster email, attendee's email, description
8. Press the confirm button
9. The event will be added into attendee's google calendar and invitation will be sent to attendee

Xref: Section 3.2.2, Create meeting

Use case: **Update Meeting Title**
Diagram:



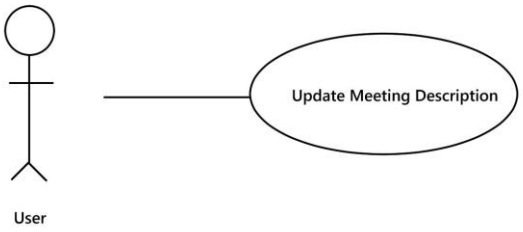
Brief Description
User updates the meeting's title

Initial Step-By-Step Description
Before this use case can be initiated, the User has already accessed the main page of the Booking Interface.

1. Select My meeting that user can browse themselves meeting there
2. UI will list the meetings that the user participant
3. Modify the event
4. UI will show the event's info
5. Modify the event's title
6. Press confirm button
7. The changing in the Book System will also apply to attendee's Google calendar

Xref: Section 3.2.3, Update Meeting title

Use case: **Update Meeting Description**
Diagram:



Brief Description
User updates the meeting's description

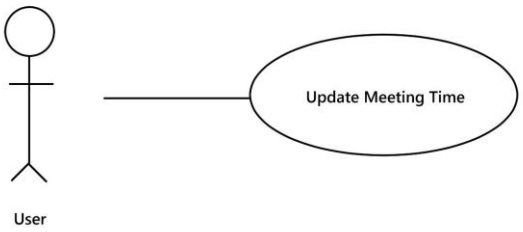
Initial Step-By-Step Description
Before this use case can be initiated, the User has already accessed the main page of the Booking Interface.

1. Select My meeting that user can browse themselves meeting there

2. UI will list the meetings that the user participant
3. Modify the event
4. UI will show the event's info
5. Modify the event's description
6. Press the confirm button
7. The changing in the Book System will also apply to attendee's Google calendar

Xref: Section 3.2.4, Update Meeting Description

Use case: **Update Meeting Time**
Diagram:



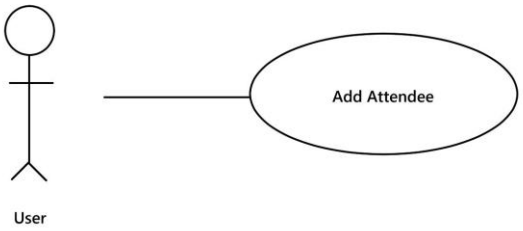
Brief Description
User updates the meeting's time

Initial Step-By-Step Description
Before this use case can be initiated, the User has already accessed the main page of the Booking Interface.

1. Select My meeting that user can browse themselves meeting there
2. UI will list the meetings that the user participant
3. Modify the event
4. UI will show the event's info
5. Modify the event's time
6. Press the confirm button
7. The changing in the Book System will also apply to attendee's Google calendar

Xref: Section 3.2.5, Update Meeting Time

Use case: **Add Attendee**
Diagram:



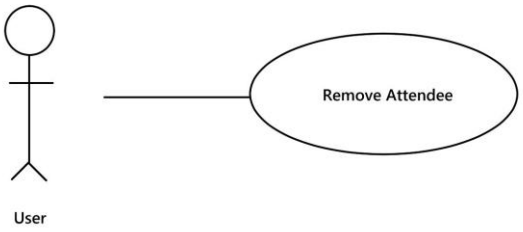
Brief Description
User add a new attendee to the event

Initial Step-By-Step Description
Before this use case can be initiated, the User has already accessed the main page of the Booking Interface.

1. Select My meeting that user can browse themselves meeting there
2. UI will list the meetings that the user participant
3. Modify the event
4. UI will show the event's info
5. Enter new attendee's email
6. Press the add button
7. Press the confirm button
8. The change in the Book System will also apply to attendee's Google calendar, and invitations will be sent to new attendees.

Xref: Section 3.2.6, Add Attendee

Use case: **Remove Attendee**
Diagram:



Brief Description
User remove the meeting in the meeting room

Initial Step-By-Step Description
Before this use case can be initiated, the User has already accessed the main page of the Booking Interface.

1. Select My meeting that user can browse themselves meeting there

2. UI will list the meetings that the user participant
3. Modify the event
4. UI will show the event's info
5. Drop down menu and select the attendee who you want to remove
6. Press the delete button
7. Press the confirm button
8. The change in the Book System will also apply to attendee's Google calendar.

Xref: Section 3.2.7, Remove Attendee

Use case: **Delete Meeting**
Diagram:

Brief Description
User remove a event

Initial Step-By-Step Description
Before this use case can be initiated, the User has already accessed the main page of the Booking Interface.

1. Select My meeting that user can browse themselves meeting there
2. UI will list the meetings that the user participant
3. Modify the event
4. UI will show the event's info
5. Press the remove button
6. The event will be removed into attendee's google calendar

Xref: Section 3.2.8, Delete Meeting

3. User Interface Specifications

Users need to be able to use button, pull-down menu, slider, textbox and other basic software operate skills.

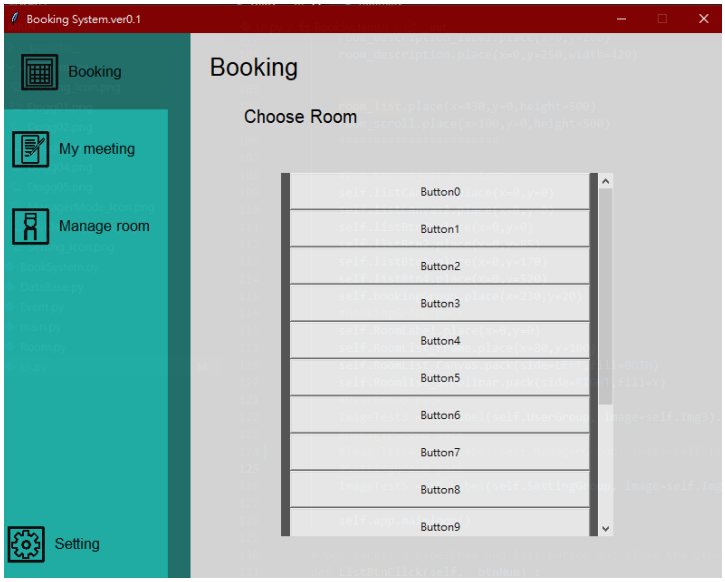


Figure 2 - User Interface

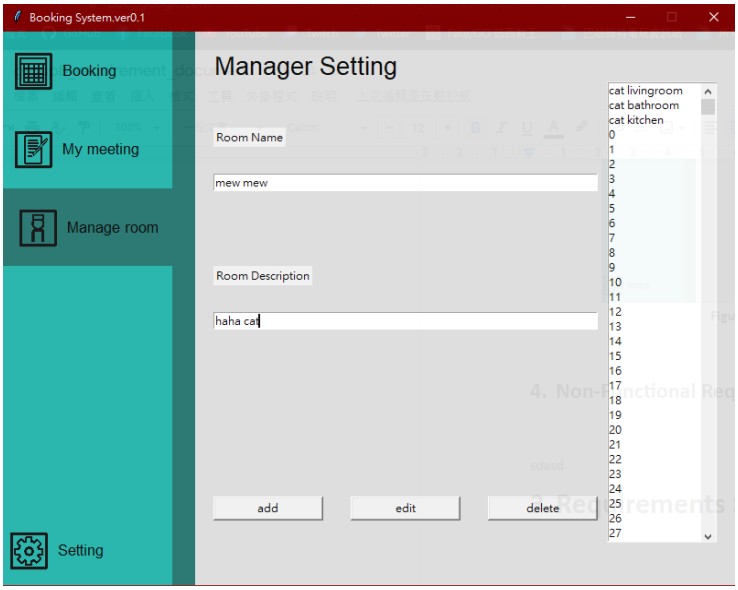


Figure 3 - Manager Interface

4. Non-Functional Requirements

Using the PC platform to execute system executable files, the system will use Google calendar for event management, so both users and administrators need to have a google account.

3. Requirements Specifications

1. External Interface Requirements

Our software will link to 2 external systems. One is the MySQL database. We use it to save data, such as meeting room data, meeting data, and meeting attendees. Another one is Google calendar API, we use this API to modify events in users' Google calendars.

2. Functional Requirements

1. Create Meeting Room

Use case name: Create meeting room

Trigger: System Manager create a meeting room

Precondition: System Manager enters the main UI.

Basic path: 1. Use Google calendar API to create a new calendar as a meeting room.

2. API will respond an ID of the calendar.

3. Using MySQL database to save calendar info.

Postcondition: The meeting room has been created.

Exception paths: System Manager may abandon the operation at any time.

Other: The calendar information includes name and meeting info (empty when created).

2. Create Meeting

Use case name: Create meeting

Trigger: User creates a meeting in the meeting room.

Precondition: User enters the meeting room UI.

Basic path: 1. Use find_calendar function to find which calendar user wants to add a meeting.

2. Function will response an ID of the calendar.

3. Create a meeting in the calendar it returned.

4. Use MySQL database to save meeting info in meeting room

Postcondition: The meeting has been created.

Exception paths: Users may abandon the operation at any time.

Other: The meeting information include name, title, description, attendees, start time and end time.

3. Update Meeting Title

Use case name: Update meeting title

Trigger: User updates title of the meeting

Precondition: User enters meeting UI.

Basic path: 1. Use find_calendar function to find calendar.

2. Use find_event function to find event that user wants to be updated.

3. Update meeting title in calendar.

4. Update the meeting information in MySQL database.

Postcondition: The title of the meeting has been updated.

Exception paths: Users may abandon the operation at any time.

4. Update Meeting Description

Use case name: Update meeting description

Trigger: User updates description of the meeting

Precondition: User enters meeting UI.

Basic path: 1. Use find_calendar function to find calendar.

2. Use find_event function to find event that user wants to be updated.

3. Update meeting description in calendar.

4. Update the meeting information in MySQL database.

Postcondition: The description of the meeting has been updated.

Exception paths: Users may abandon the operation at any time.

5. Update Meeting Time

Use case name: Update meeting time

Trigger: User updates time of the meeting.

Precondition: User enters meeting UI.

Basic path: 1. Use find_calendar function to find calendar.

2. Use find_event function to find the event that the user wants to be updated.

3. Update meeting time in calendar.

4. Update the meeting information in MySQL database.

Postcondition: The time of the meeting has been updated.

Exception paths: Users may abandon the operation at any time.

6. Add Attendee

Use case name: add attendee

Trigger: User adds attendee of the meeting.

Precondition: User meeting UI.

Basic path: 1. Use find_calendar function to find calendar.

2. Use find_event function to find events that the user wants to add attendees into.

3. Update meeting attendee in calendar.

4. Update the meeting information in MySQL database.

Postcondition: The attendee has been added into the meeting.

Exception paths: Users may abandon the operation at any time.

Other: Attendee information includes Email only.

7. Remove Attendee

Use case name: Remove attendee

Trigger: User removes attendee of the meeting.

Precondition: User enters the meeting UI.

Basic path: 1. Use find_calendar function to find calendar.

2. Use find_event function to find the event.

3. Use find_attendee function to find attendee to be removed.

4. Update meeting information in calendar.

5. Update the meeting information in MySQL database.

Postcondition: The attendee has been removed from the meeting.

Exception paths: Users may abandon the operation at any time.

Other: Attendee information includes Email only.

8. Delete Meeting

Use case name: Delete meeting

Trigger: User Deletes the meeting.

Precondition: User enters meeting UI.

Basic path: 1. Use find_calendar function to find calendar.

2. Use find_event function to find the event that the user wants to delete.

3. Use Google calendar API to delete the event.

4. Remove the meeting information in MySQL database.

Postcondition: The meeting has been deleted.

Exception paths: Users may abandon the operation at any time.

9. Delete Meeting Room

Use case name: Delete meeting room

Trigger: System Manager deletes the selected meeting room.

Precondition: System Manager enters the meeting room UI.

Basic path: 1. Use find_calendar function to find the calendar that the System Manager wants to delete.

2. Use Google calendar API to delete calendar.

3. Remove all the meeting room information includes all events in the calendar in MySQL database.

Postcondition: The meeting room has been deleted.

Exception paths: System Manager may abandon the operation at any time

4. Other Nonfunctional Requirements

1. Performance Requirements

Make sure the database won't be stuck when many users are using the program at the same time. Maybe need load balance if too many users access the database at the same time.

2. Safety Requirements

We need to try our best to prevent users from doing irreversible or illegal operations, such as delete the database or access others data. Besides, it's better to back-up the database once in a while.

3. Security Requirements

We need to protect the client and the meeting’s information. There are some ways we can do this, one is to encrypt the data we transfer, another is to verify the user's identity before we access the database.

Software Architecture 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

Revision History

Name	Data	Reason for Change	Version
Draft	2021/12/14	First version	1.0

1. Introduction

I. Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

II. Scope

This Software Architecture Document applies to the Booking System.

III. 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

IV. References

- Software Architecture Document
- Artifact_Software Architecture Document

2. Architectural Representation

This document presents the architectural as a series of views; use case view, process view, deployment view, and implementation view. These views are presented as Rational Rose Models and use the Unified Modeling Language (UML).

3. Architectural Goals and Constraints

We hope that the system can be used in the college for teachers and students to schedule meetings, and also we hope that our system can be very flexible, so we divide our system into four parts, and it will be introduced later.

Our system can be used in any place that has an internet connection, we will make sure that the system, database, google calendar are synchronized.

When we are developing our system, we need to follow a certain coding style, it could make our team members easier to understand each other's code and maintain.

We use python as our main programming language, because python has a very strong community, so we can use many APIs and libraries, and the most important thing is that we can search information and solutions easily.

We have four people in our team, **Hsieh Jun-Yao** as Project Manager and developer, **Liao Sheng-Hao** as Program Manager and developer, **Pu Chi-Hao** as UI designer and developer, **Huang Chen-En** as developer, and we have a

meeting every Tuesday.

4. Use-Case View

A description of the use-case view of the software architecture. The Use Case View is an important input to the selection of the set of scenarios and/or use cases that are the focus of an iteration. It describes the set of scenarios and/or use cases that represent some significant, central functionality. It also describes the set of scenarios and/or use cases that have a substantial architectural coverage (that exercise many architectural elements) or that stress or illustrate a specific, delicate point of the architecture.

The use cases in this system are listed below. Use cases in **bold** are significant to the architecture. A description of these use cases can be found later in this section.

- **Add Room**
- **Delete Room**
- View Event
- **Add Event**
- **Delete Event**
- **Modify Event**
- Add Calendar
- Delete Calendar
- Add Event to Google Calendar
- Delete Event at Google Calendar
- Modify Event at Google Calendar
- Setting Interface

The following diagrams depict the use cases in the system.

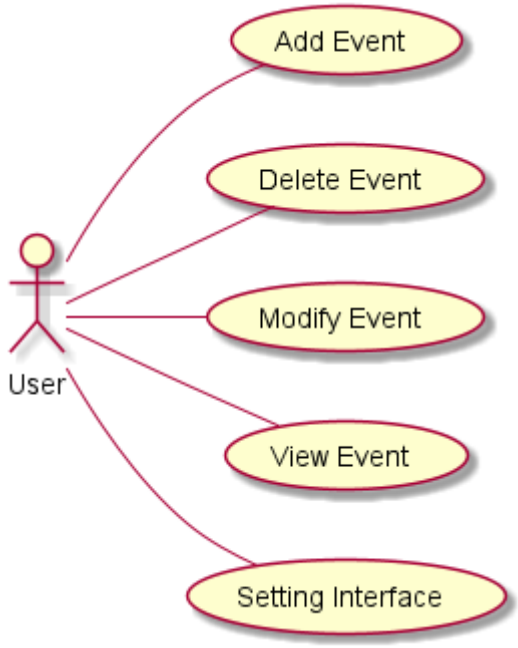


Figure 1 - User Use Cases

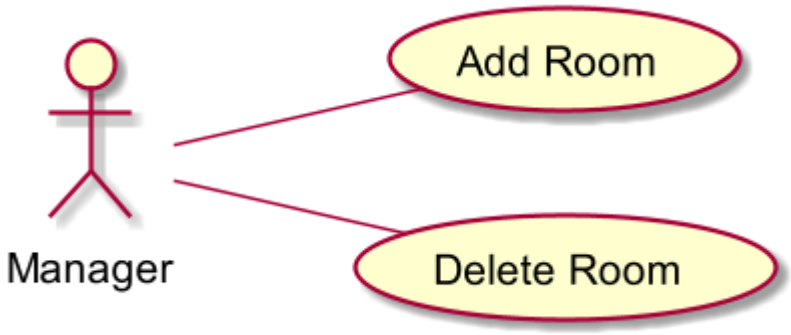


Figure 2 - Manager Use Cases

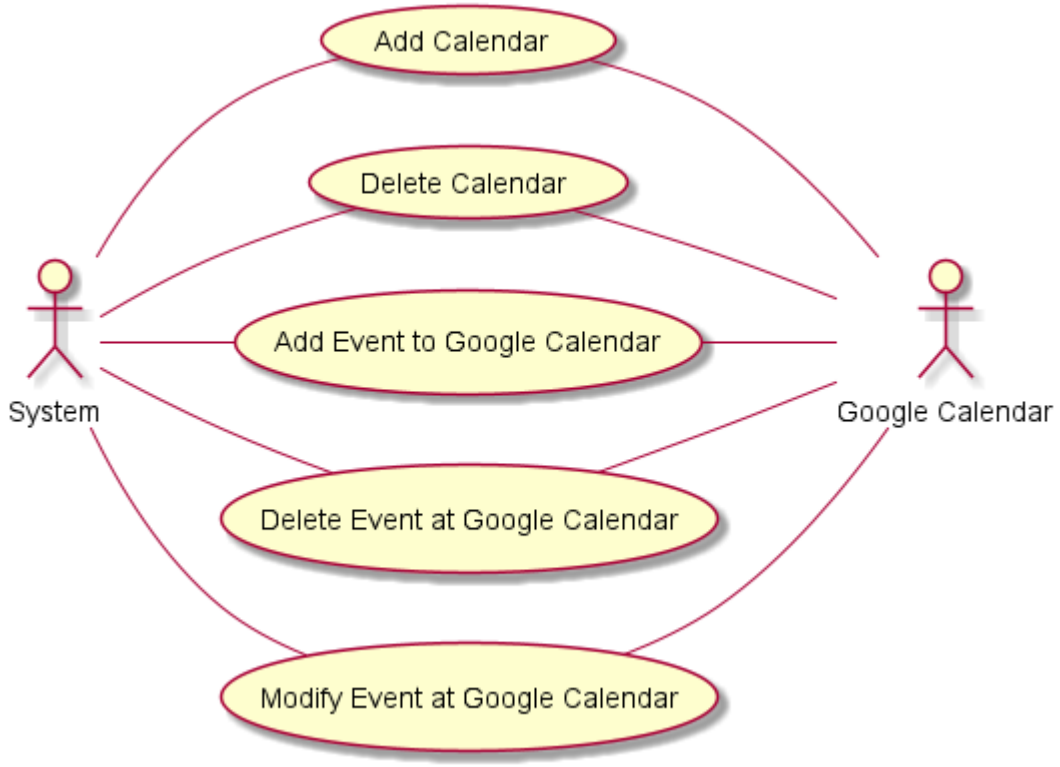


Figure 3 - System Use Cases

Significant Use Case Descriptions

1. Add Room
   When Manager adds a new room, this use case will appear
2. Delete Room
   When Manager deletes a room, this use case will appear
3. View Event
   The user case appears when the user wants to check the events of the specified room and time.
4. Add Event
   This use case will appear when the User wants to add an event.
5. Delete Event
   This use case will appear when the User wants to delete an event.
6. Modify Event
   This use case will appear when the User wants to modify an event.。
7. Add Calendar
   System adds a new calendar in google calendar, this use case will appear.
8. Delete Calendar
   System deletes a calendar in google calendar, this use case will appear.
9. Add Event to Google Calendar
   System adds a new event in google calendar, this use case will appear.
10. Delete Event at Google Calendar
    System deletes a new event in google calendar, this use case will appear.
11. Modify Event at Google Calendar
    System modifies a new event in google calendar, this use case will appear.

5. Logical View

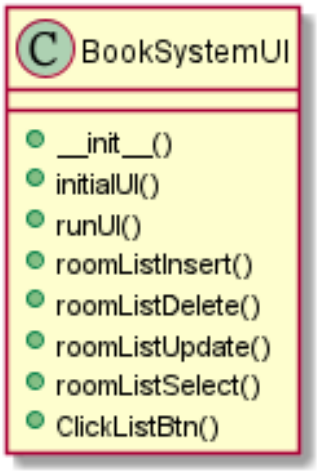
I. Overview

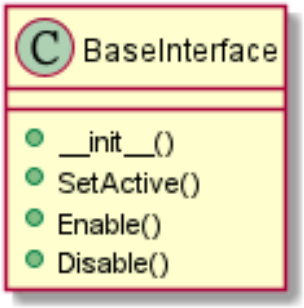
The logical view of the Booking System is comprised of 5 main packages:

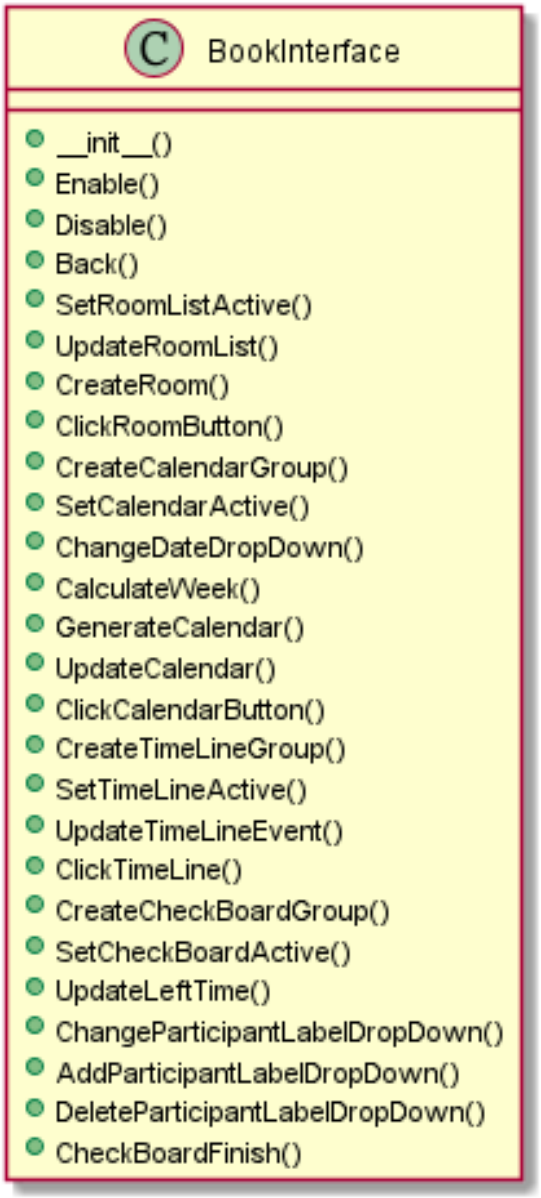
- Presentation
  - contain classes for user interface
- Application
  - contain classes for core function
- Domain
  - contain classes for storing information/content
- Persistence
  - contains classes to persist specific objects within the system.
- Services
  - contains classes to provide system-level classes for maintenance purposes

II. Architecturally Significant Design Packages

Presentation package

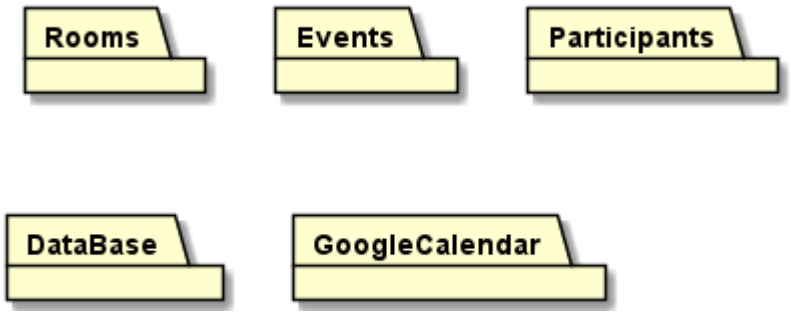

BookSystemUI
__init__()
initialUI()
runUI()
roomListInsert()
roomListDelete()
roomListUpdate()
roomListSelect()
ClickListBtn()


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()

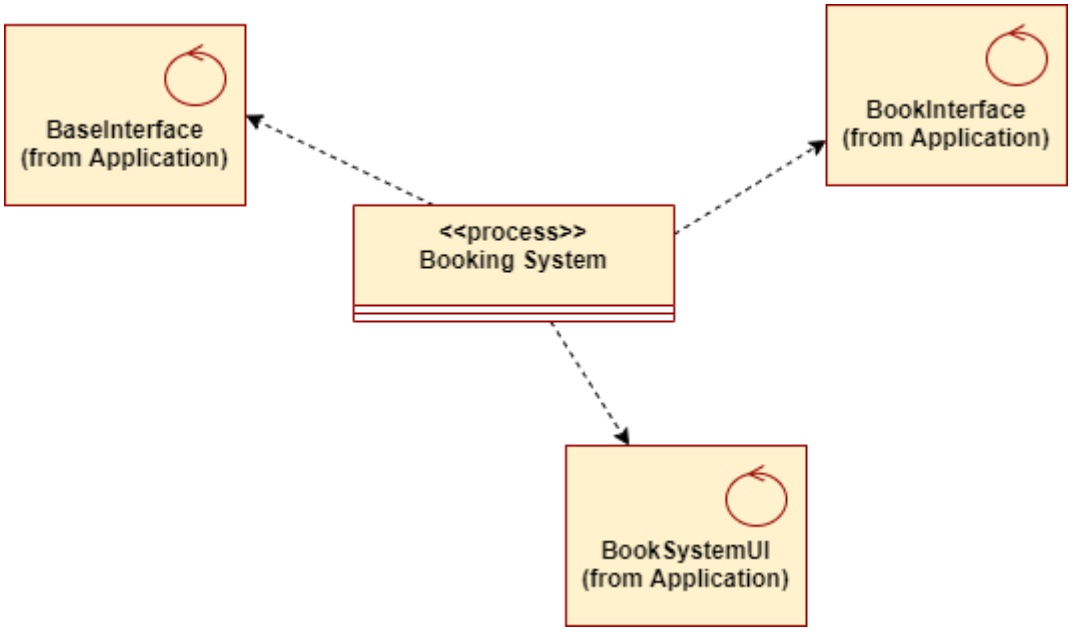
Application package

Domain package



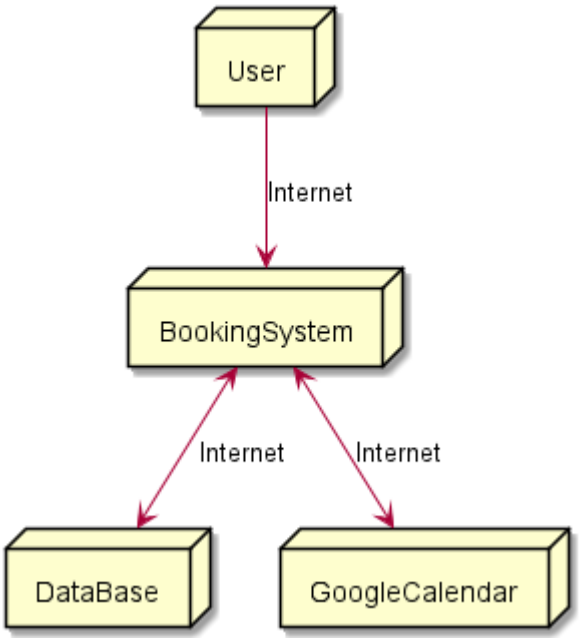
6. Process View

In the previous section we mentioned application functions, then we designed a single process to provide server functions for the Booking System. Threads for application functions are part of this process.
The process diagram of the system was made and can be viewed as follows:



7. Development View

As previously mentioned, the system can be used in any place that has an internet connection. We use MySQL as our database.



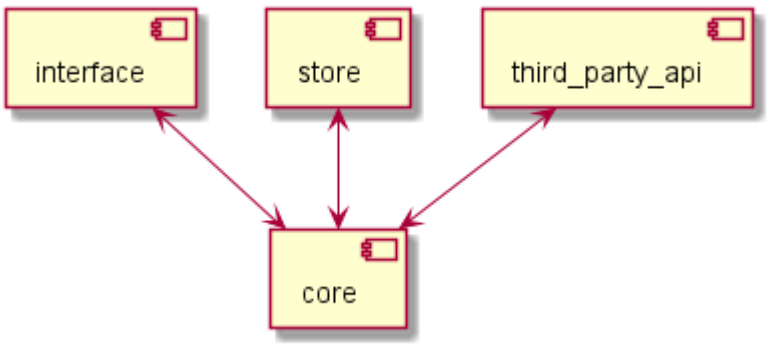
8. Implementation View

I. Overview

Booking system contains 4 layers.

- core layer
  - Most common and important features.
  - Every action will pass through this layer
- interface layer
  - interface to communicate with user
- store layer
  - store information/content into database
- third party api layer
  - connect to third party service through api

II. Layers



9. Size and Performance

Because Google Calendar API limits, we have 1000000 queries per day, it will cause error if more than limited queries per day.

10. Quality

The above software supports running in windows 10 environment and provides a graphical interface.

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

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

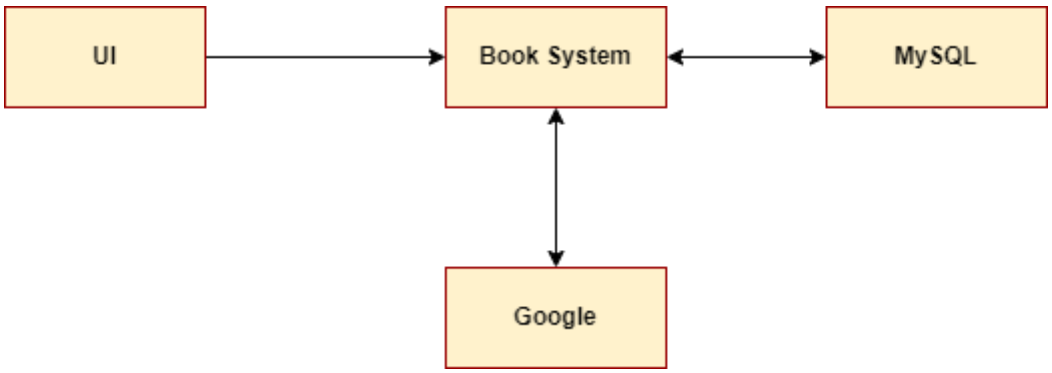
- example_design_SDD
- SDDTemplate
- Software Design Document
- Software Design Document, Testing, and Deployment and Configuration Management

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



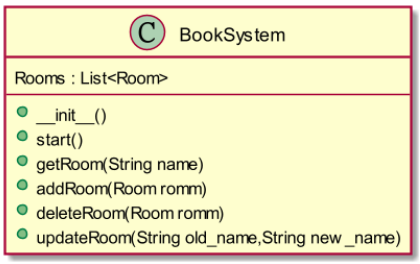
II. Decomposition Description

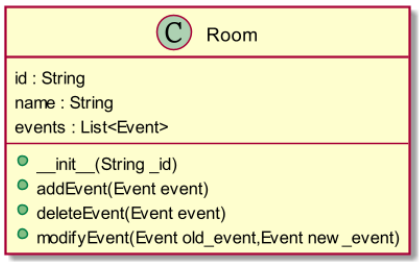
Take modify event function to make a example:

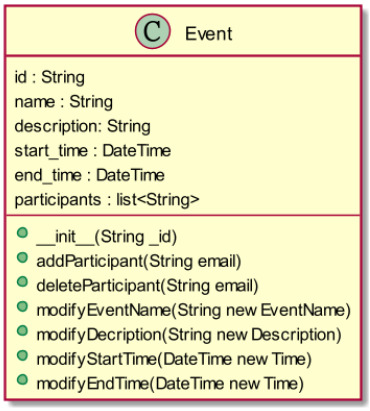
User trigger the modify event function, the BooksystemUI send request to BookInterface, then BookInterface received the request, contact BookSystem to modify, Booksystem find the meeting room and designated event, and the Event class modity event, return the new event to Booksystem and return to BookInterface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

Object diagrams

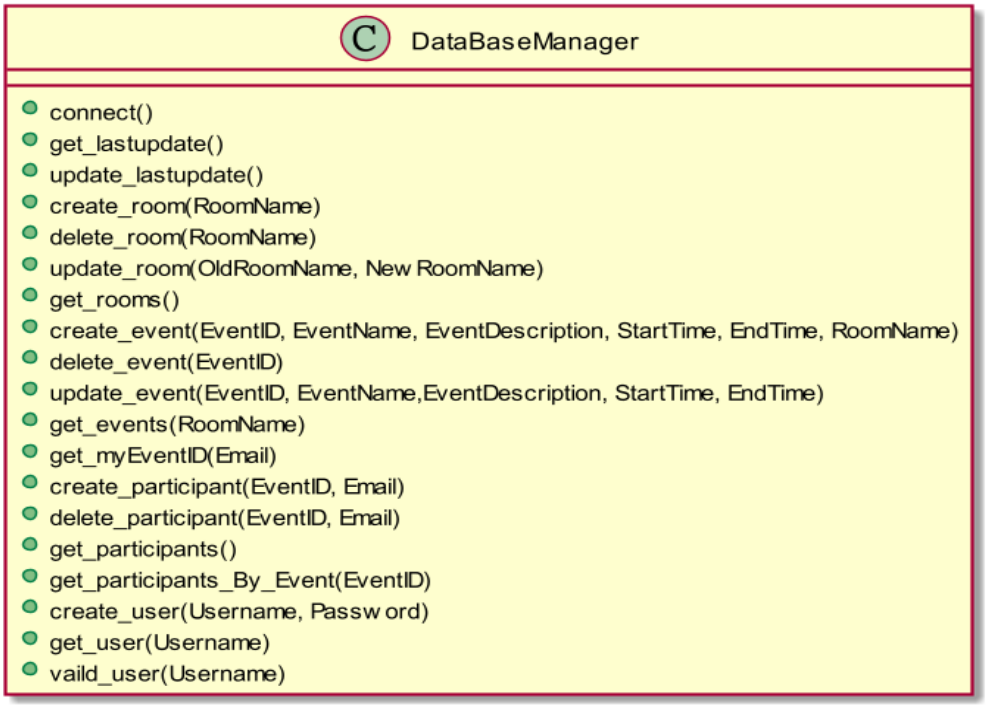
- **BookSystem**



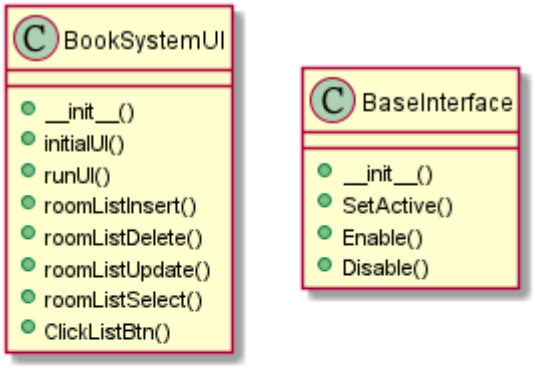


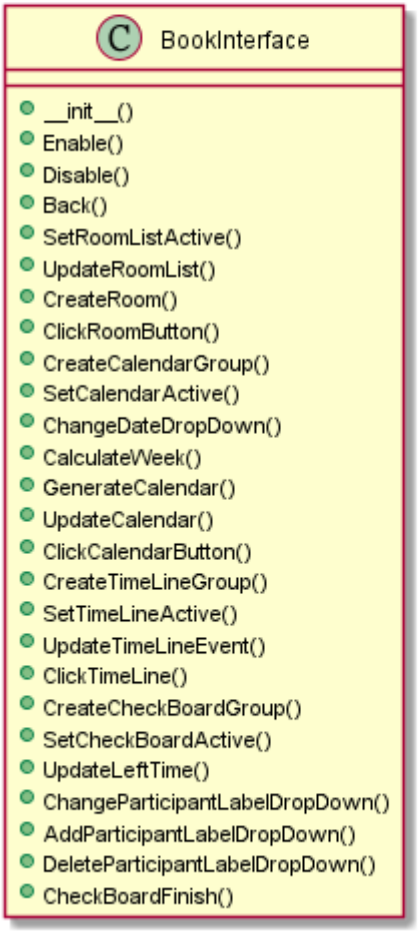


- **DataBase**

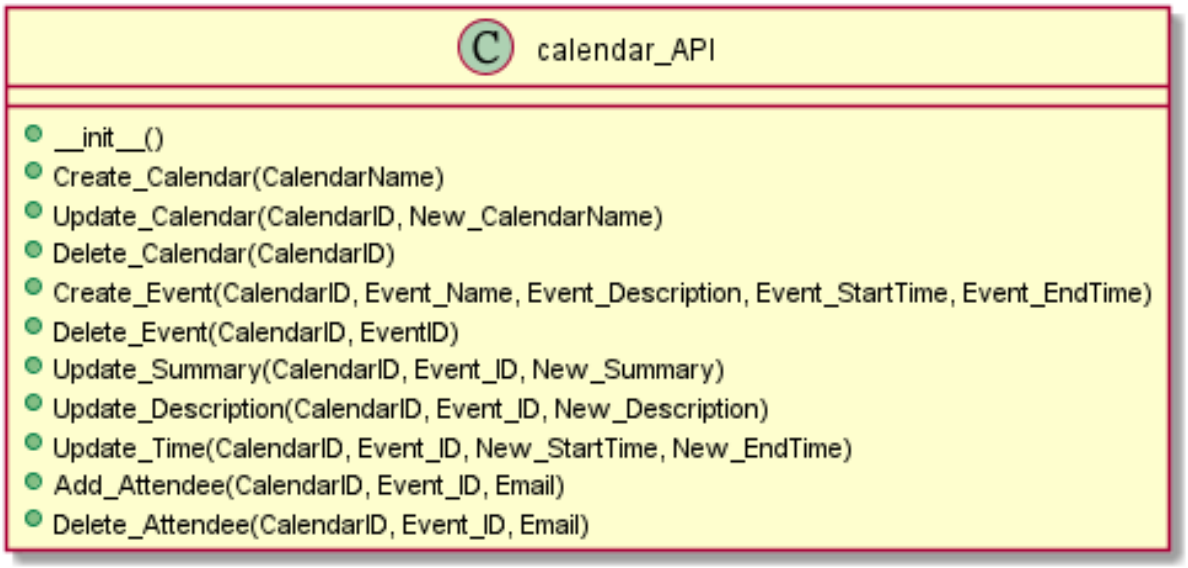


- **UI**

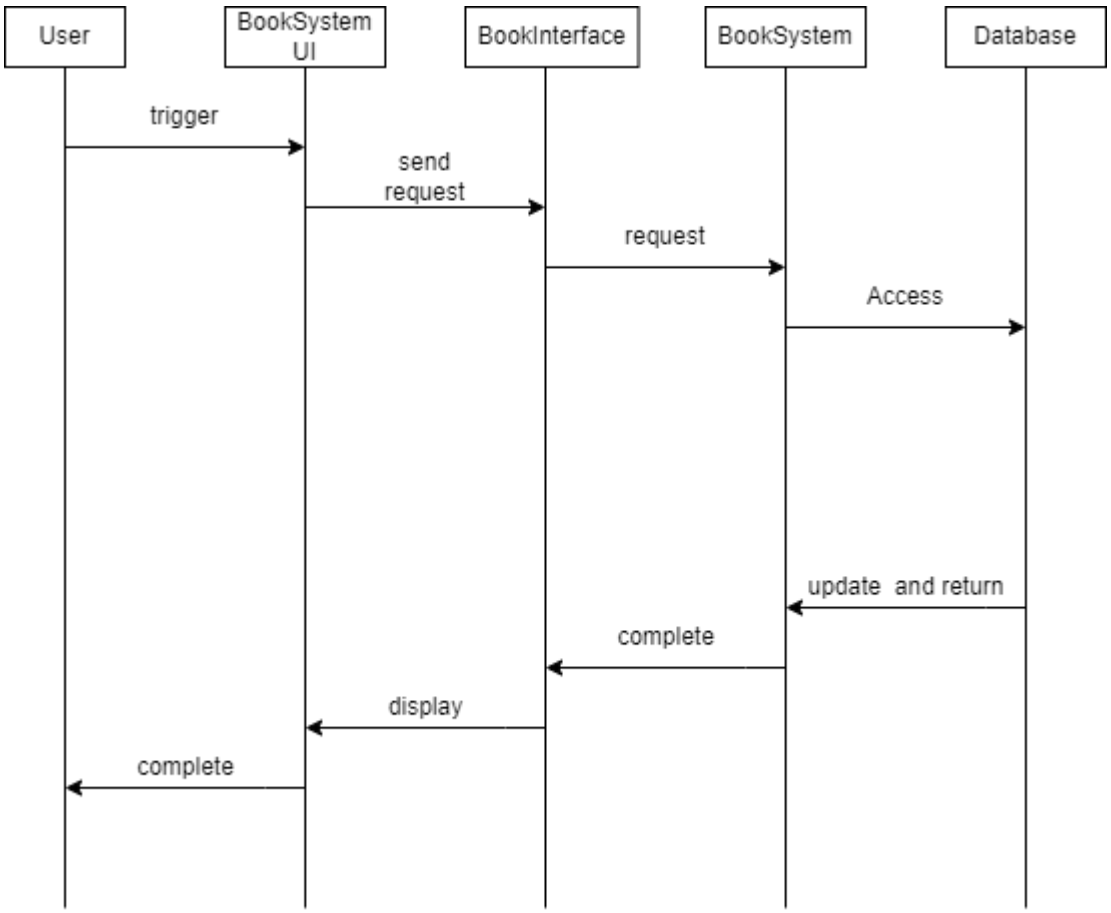

BookSystemUI
__init__()
initialUI()
runUI()
roomListInsert()
roomListDelete()
roomListUpdate()
roomListSelect()
ClickListBtn()
BaseInterface
__init__()
SetActive()
Enable()
Disable()


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()

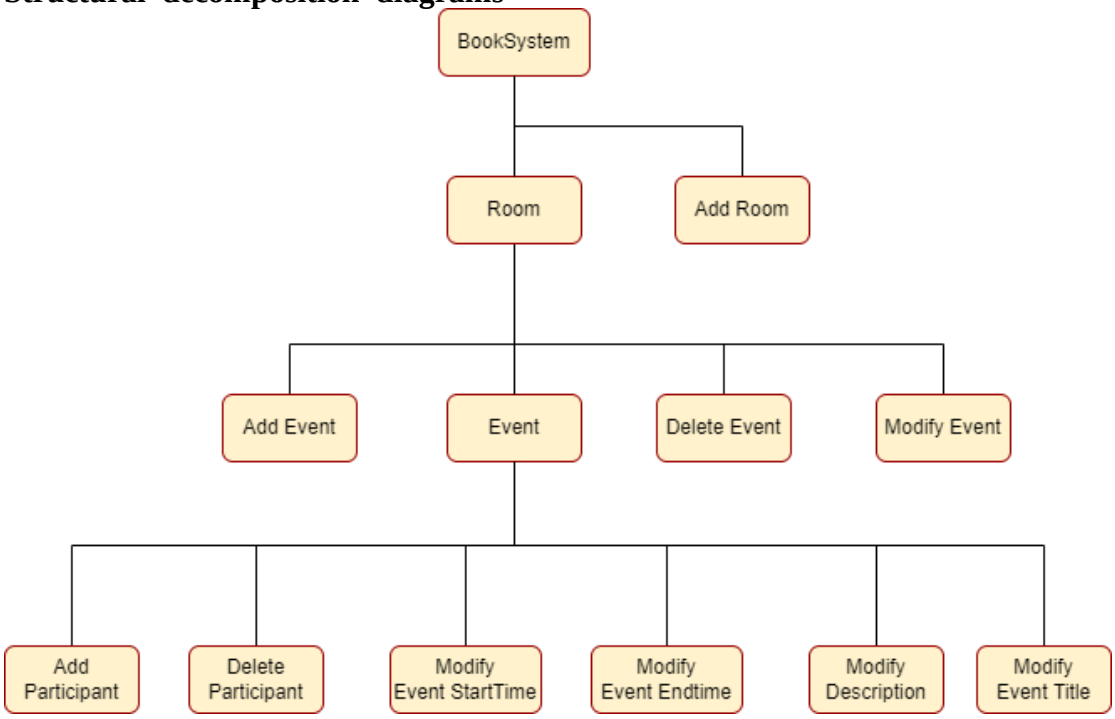
- **Google Calendar API**



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

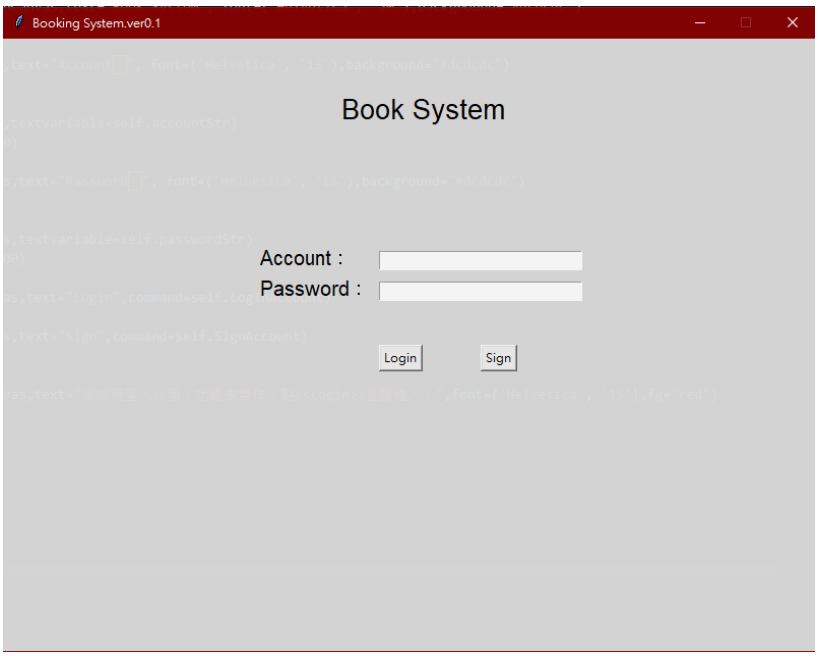


Figure 1.Login UI

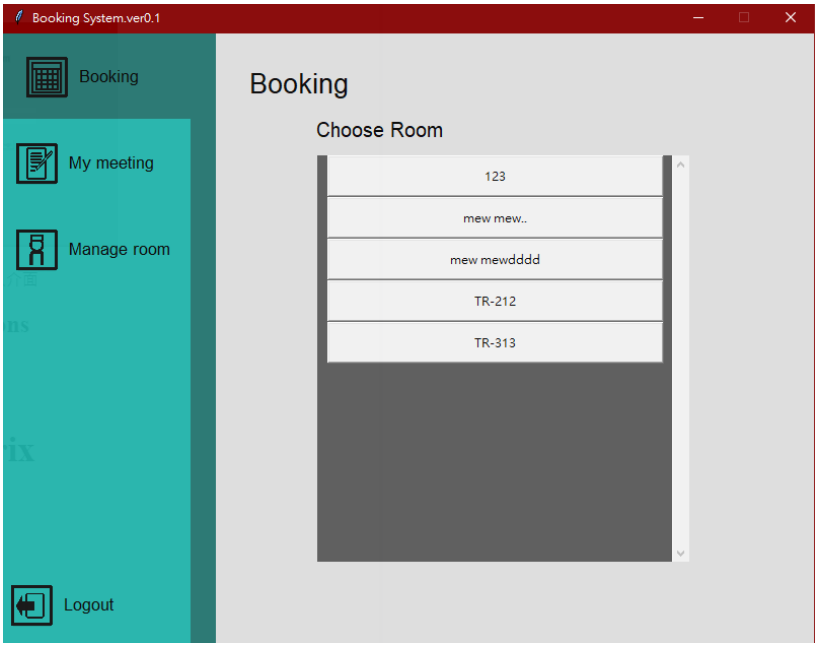


Figure 2. Room List

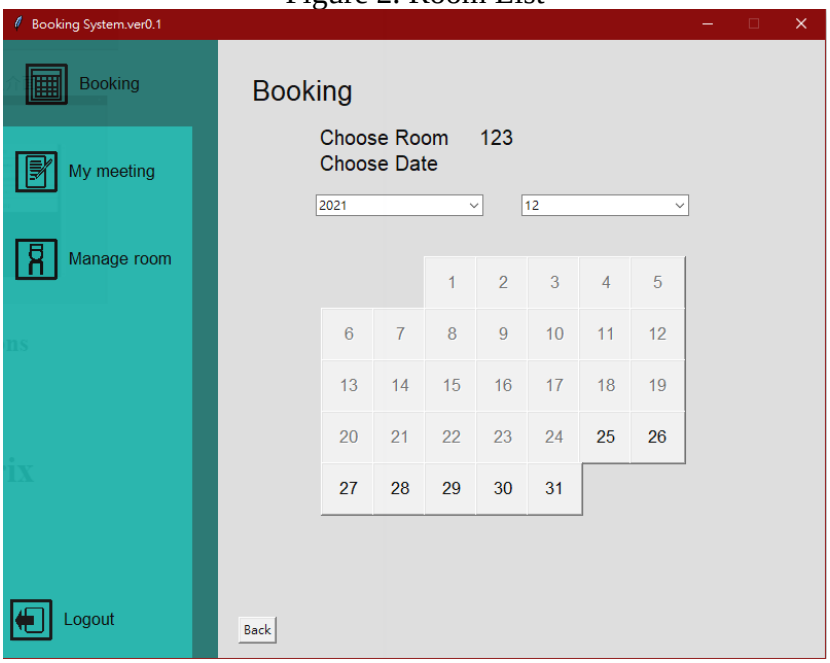


Figure 3. Calendar

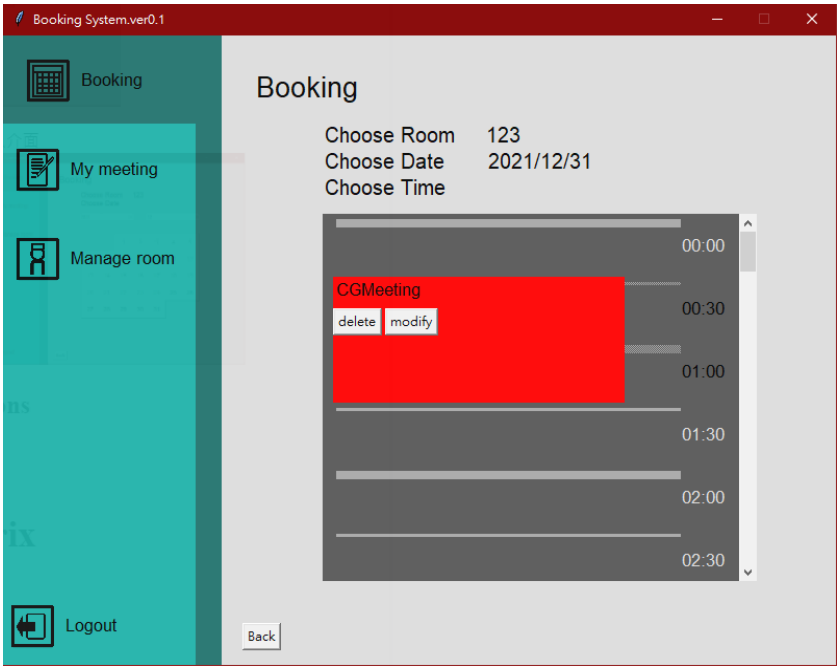


Figure 4. Timeline

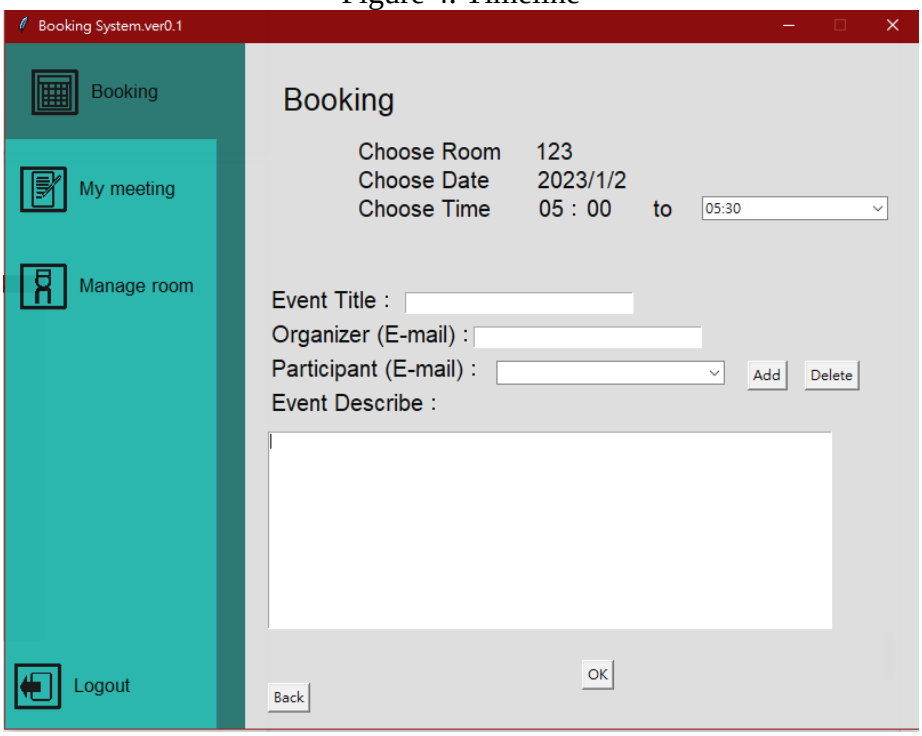


Figure 5. Booking

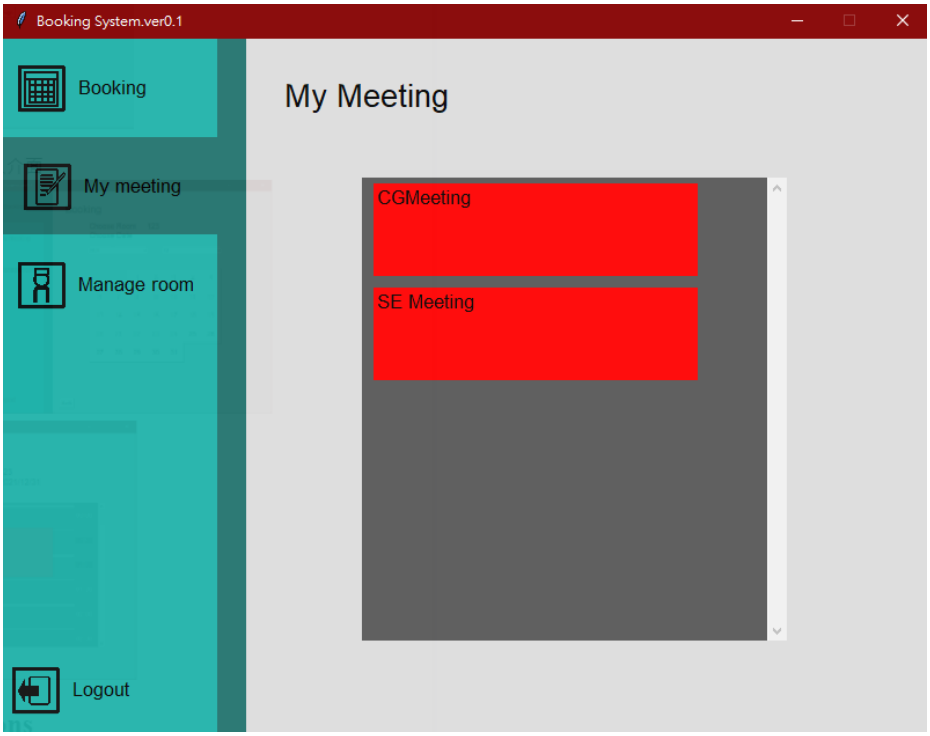


Figure 6. My Meeting

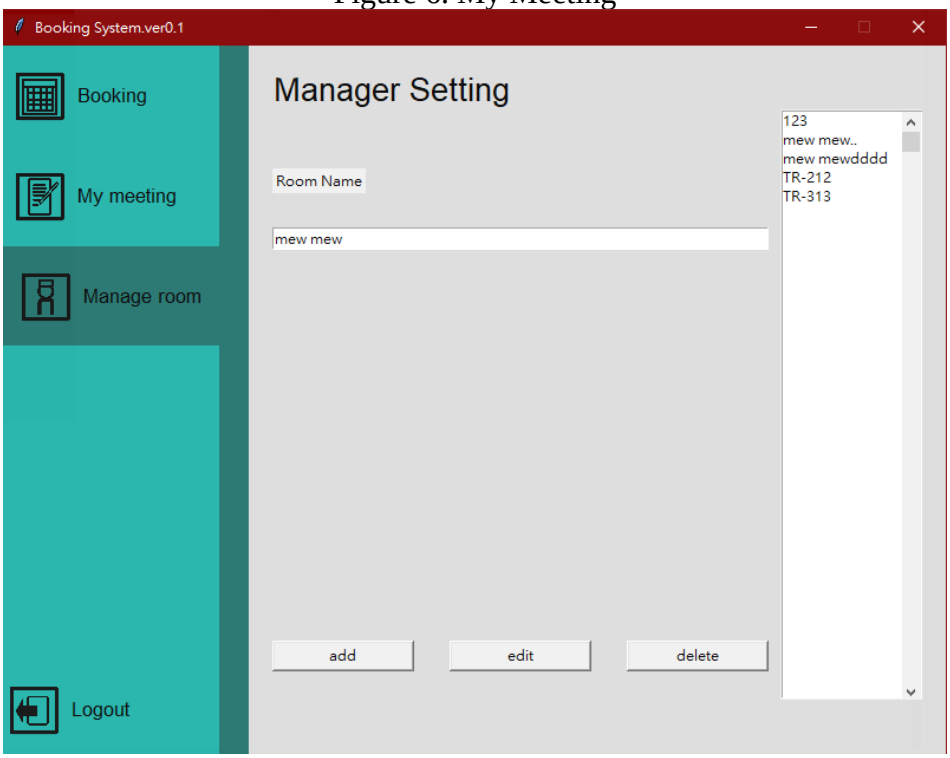
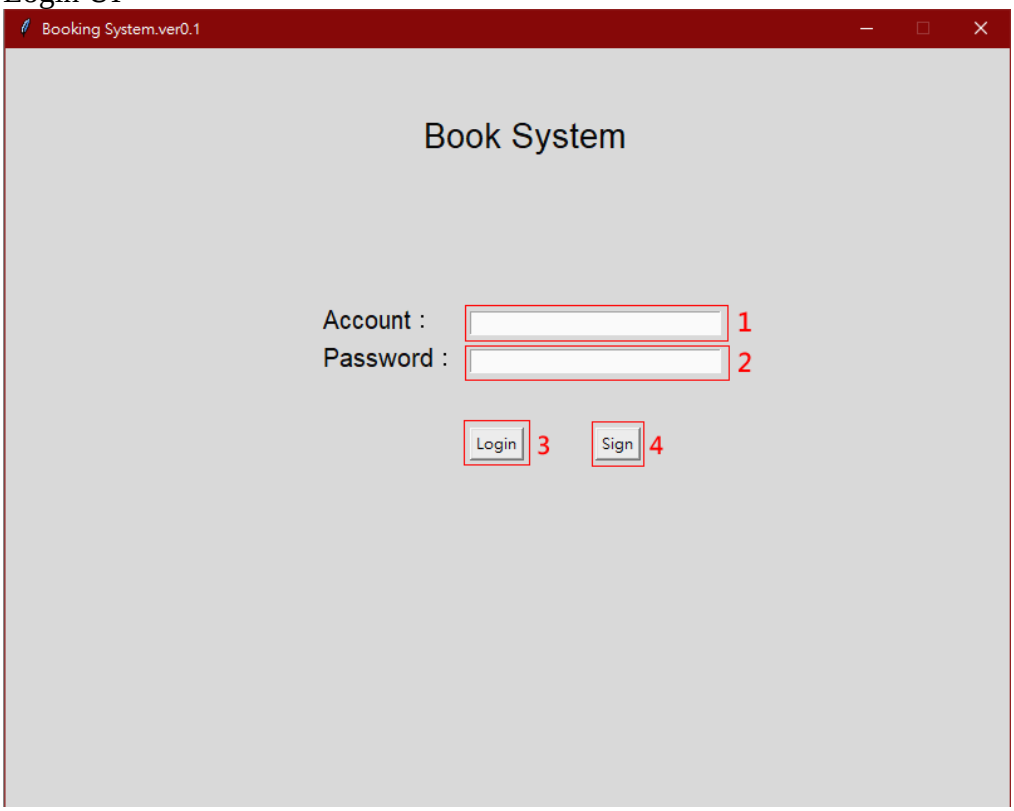


Figure 7. Manager Options

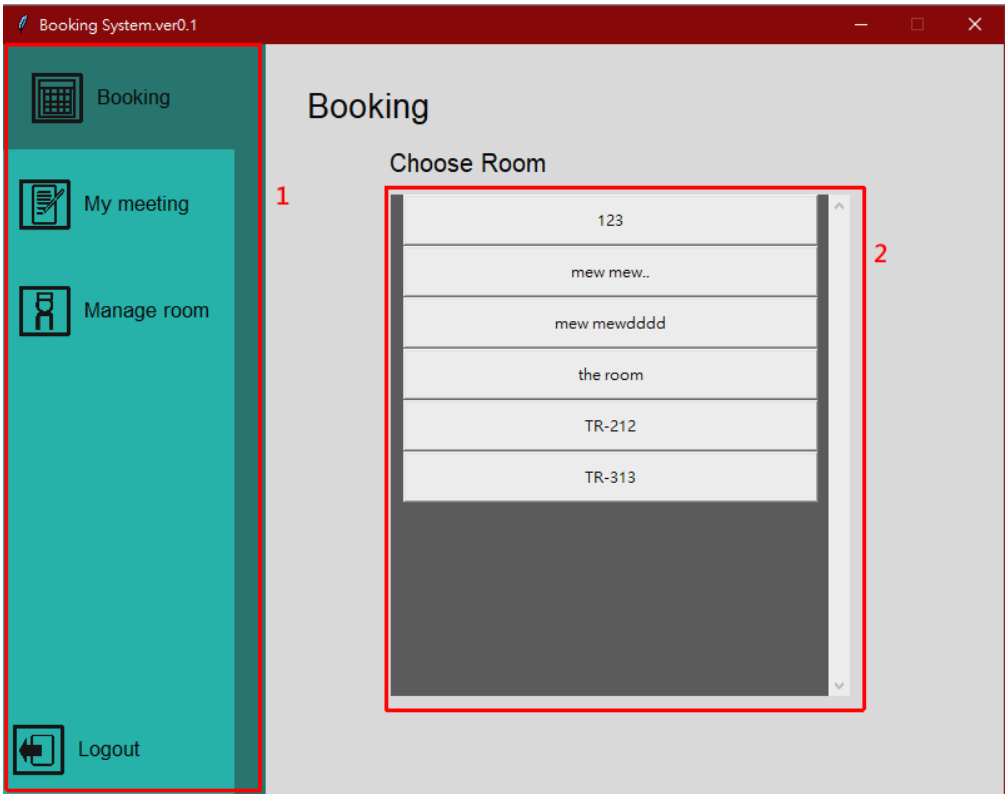
VII. Screen Object and Actions

1. Login UI



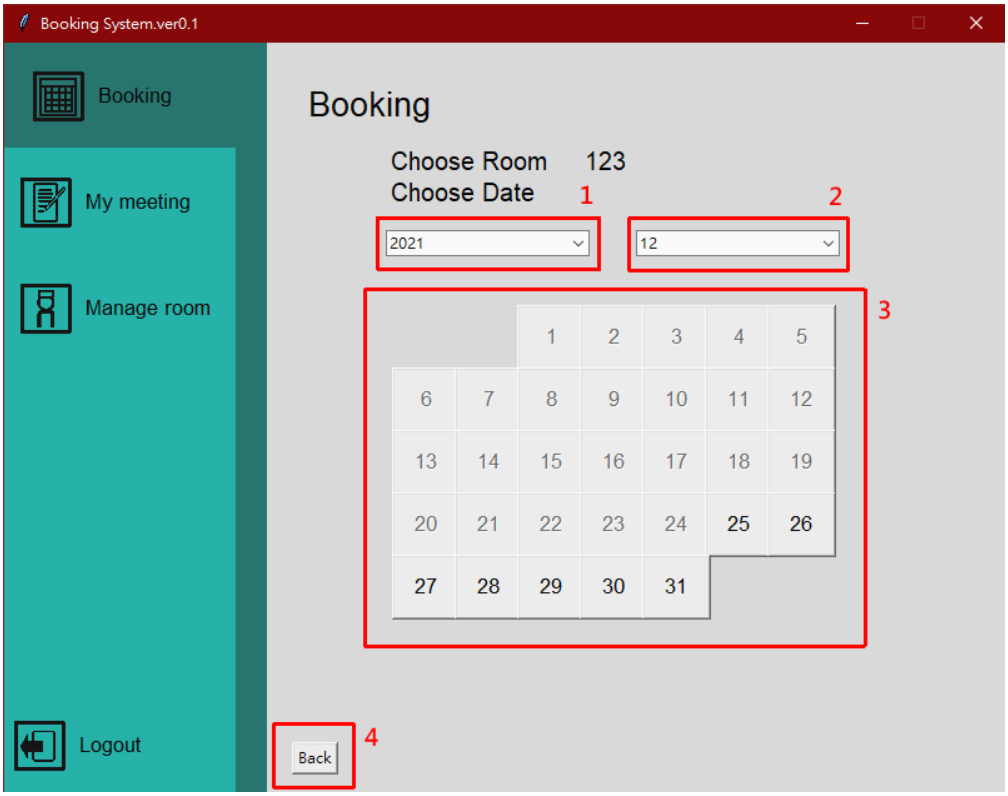
1.Account	Input Email account to register
2.Password	Input password from the account
3.Login Button	Click to login the system
4.Register Button	Click to register account the system

2. Initial UI



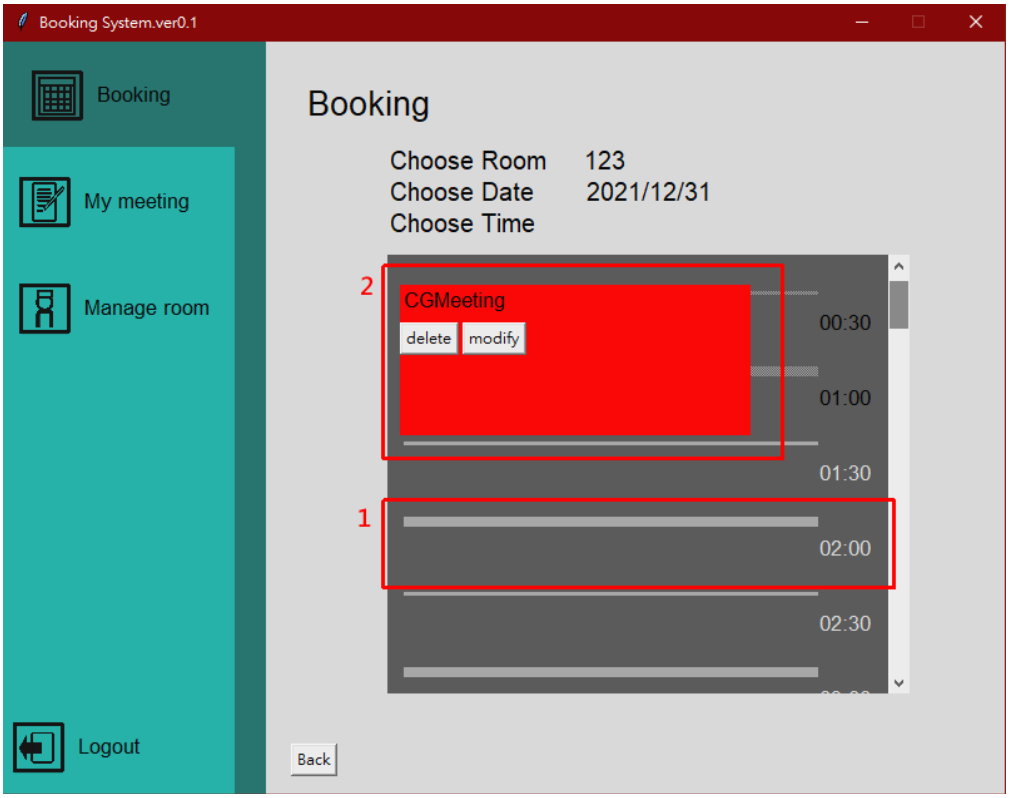
1. Function list	Through this list, you can switch to other functional interfaces
2. Meeting rooms list	List all available meeting rooms and click on them to enter the reservation process.

3. 日歷



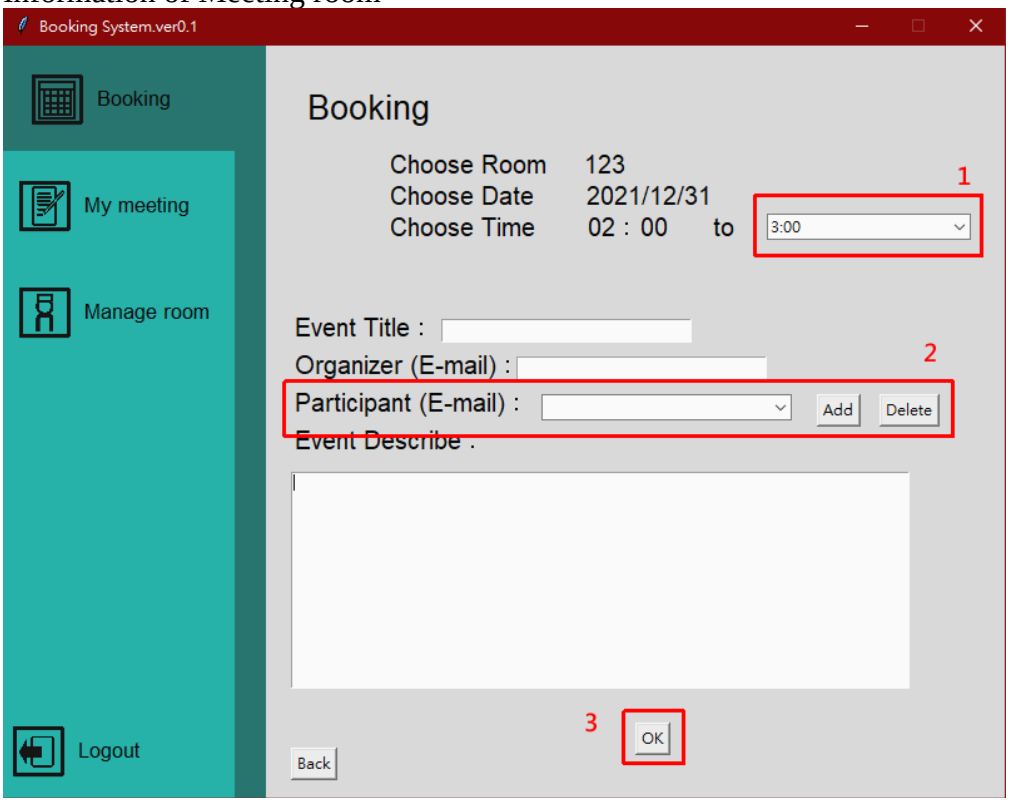
1. Dropdown(year)	Click to select the year you want to reserve
2. Dropdown(Month)	Click to select the month you want to reserve
3. Calendar	Click to select the date you want to reserve
4. Back button	Back to previous UI

4. Events Timeline



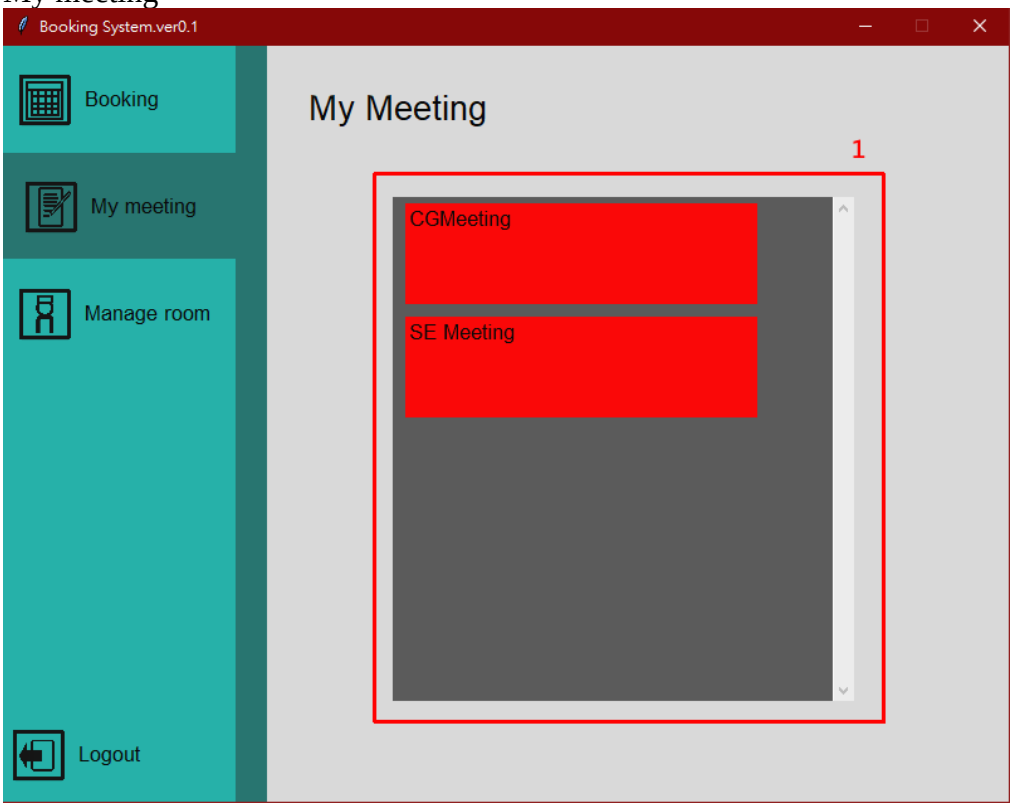
1. Timeline	Click to select the meeting start time
2. EventStamp	Timelines that have already been booked will be covered by this red mark and can be modified or deleted if the owner of the event is the owner.

5. Information of Meeting room



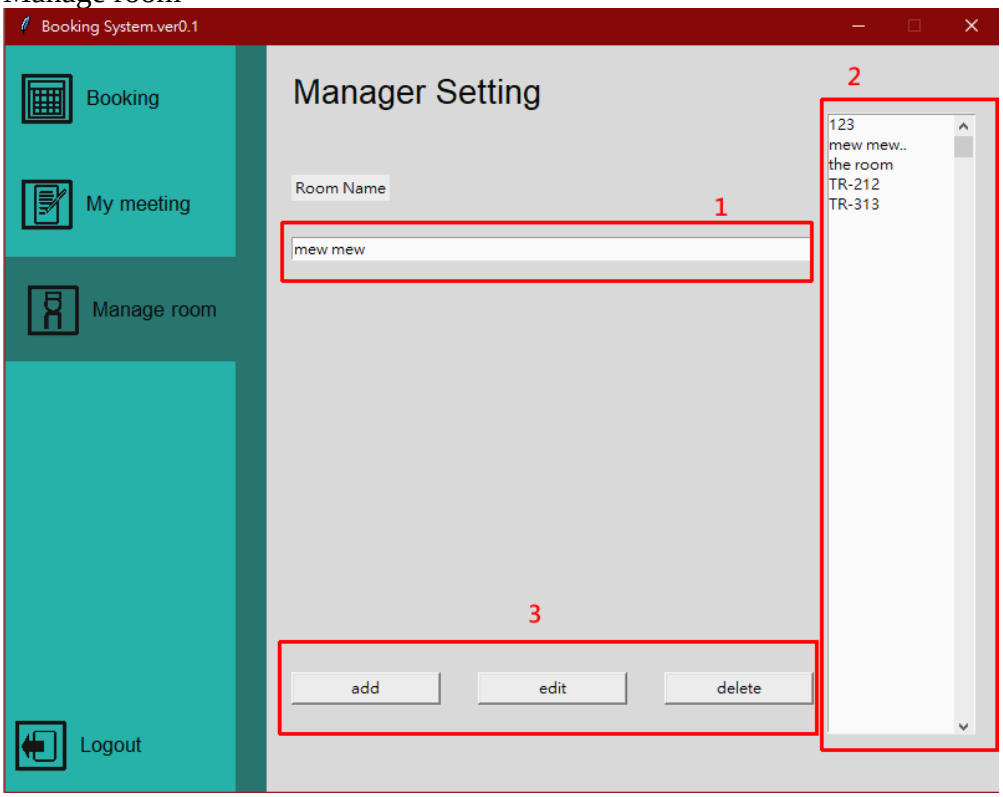
1. Dropdown(End time)	Click to select the end time of the meeting
2. Participant	After entering the e-mail address of the participant in the left column, you can add the participant through the Add button, or click the drop-down list to select an existing participant and press Delete to remove it.
3.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