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

[1. Introduction 1](#_Toc90404556)

[I. Purpose 1](#_Toc90404557)

[II. Scope 1](#_Toc90404558)

[III. Definition, Acronyms and Abbreviations 1](#_Toc90404559)

[IV. References 2](#_Toc90404560)

[2. Architectural Representation 2](#_Toc90404561)

[3. Architectural Goals and Constraints 2](#_Toc90404562)

[4. Use-Case View 3](#_Toc90404563)

[Significant Use Case Descriptions 5](#_Toc90404564)

[5. Logical View 6](#_Toc90404565)

[I. Overview 6](#_Toc90404566)

[II. Architecturally Significant Design Packages 6](#_Toc90404567)

[6. Process View 8](#_Toc90404568)

[7. Development View 9](#_Toc90404569)

[8. Implementation View 10](#_Toc90404570)

[I. Overview 10](#_Toc90404571)

[II. Layers 10](#_Toc90404572)

[9. Size and Performance 10](#_Toc90404573)

[10. Quality 11](#_Toc90404574)

Revision History

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

# Introduction

## 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.

## Scope

This Software Architecture Document applies to the Booking System.

## 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 |

## References

* [Software Architecture Document](https://moodle.ntust.edu.tw/mod/resource/view.php?id=424119)
* [Artifact\_Software Architecture Document](https://moodle.ntust.edu.tw/mod/resource/view.php?id=424118)

# 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).

# 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.

# 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.

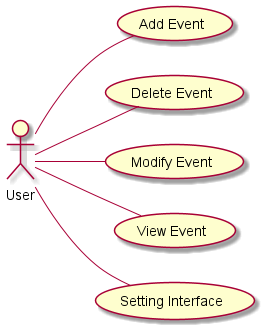


Figure 1 - User Use Cases

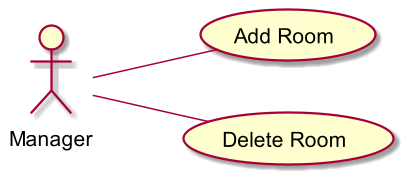


Figure 2 - Manager Use Cases

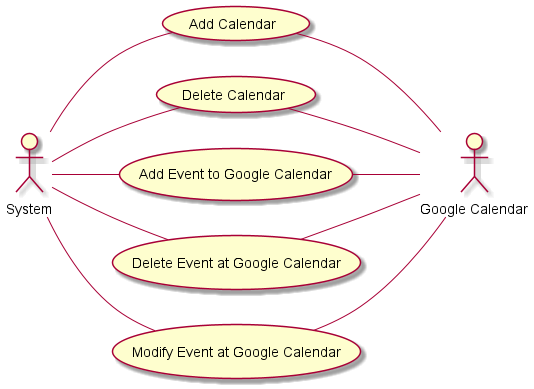


Figure 3 - System Use Cases

#### Significant Use Case Descriptions

1. Add Room

When Manager adds a new room, this use case will appear

1. Delete Room

When Manager deletes a room, this use case will appear

1. View Event

The user case appears when the user wants to check the events of the specified room and time.

1. Add Event

This use case will appear when the User wants to add an event.

1. Delete Event

This use case will appear when the User wants to delete an event.

1. Modify Event

This use case will appear when the User wants to modify an event.。

1. Add Calendar

System adds a new calendar in google calendar, this use case will appear.

1. Delete Calendar

System deletes a calendar in google calendar, this use case will appear.

1. Add Event to Google Calendar

System adds a new event in google calendar, this use case will appear.

1. Delete Event at Google Calendar

System deletes a new event in google calendar, this use case will appear.

1. Modify Event at Google Calendar

System modifies a new event in google calendar, this use case will appear.

# Logical View

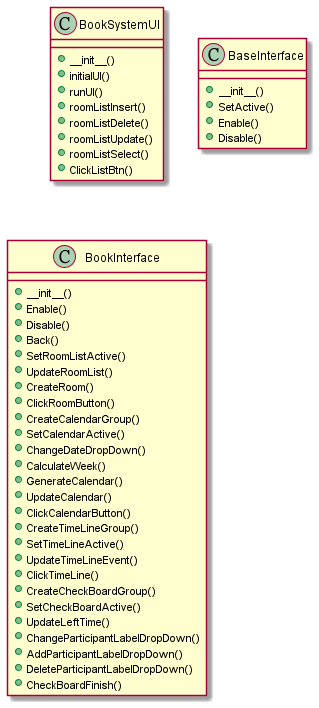
## Overview

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

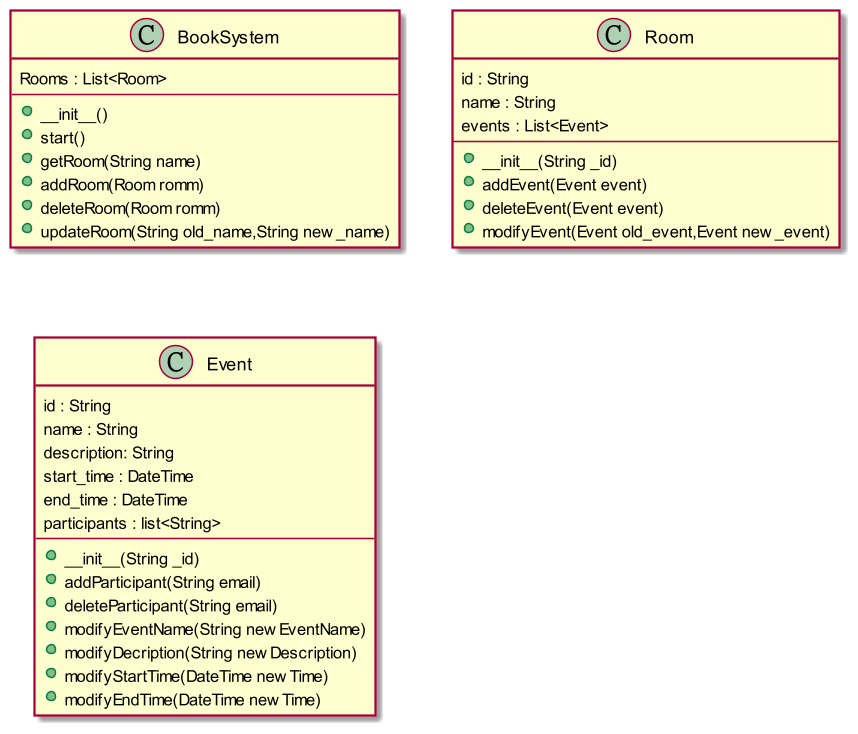
* Presentation
  + contain classes for user interface
* Application
  + contain classes for core function
* Domain
  + contain classes for storing information/content
* Persistence
  + contains classes to persist specific objects within the system.
* Services
  + contains classes to provide system-level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

## Architecturally Significant Design Packages

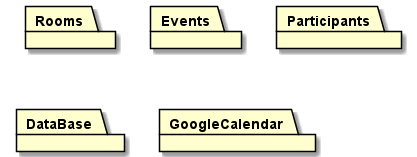
Presentation package



Application package



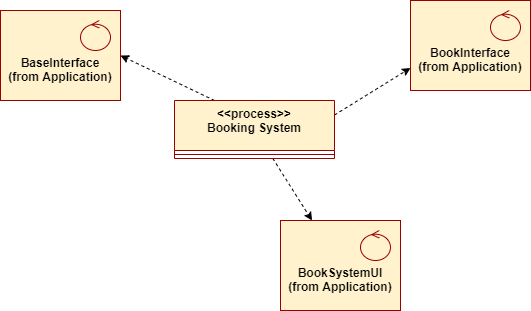
Domain package



# Process View

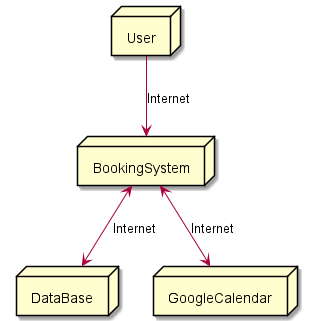
In the previous section we mentioned application functions, then we designed a single process to provide server functions for the Booking System. Threads for application functions are part of this process.

The process diagram of the system was made and can be viewed as follows:



# Development View

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



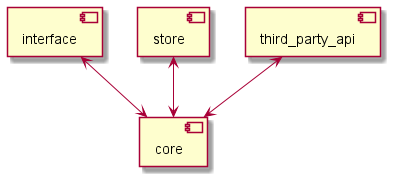
# Implementation View

## 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

## Layers



# 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.

# Quality

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