

---

COMP2432  
G06 Group Project Reoprt

---

Room Booking Manager

19081789D MAN, Furui  
19078543D WANG, Meng  
18080998D WU, Junyu  
19079008D XING, Shiji

# Contents

<b>1</b>	<b>Introdoction</b>	<b>1</b>
<b>2</b>	<b>Scope</b>	<b>2</b>
<b>3</b>	<b>Concept</b>	<b>3</b>
3.1	FCFS Scheduling . . . . .	3
3.2	PRIIO Scheduling . . . . .	3
<b>4</b>	<b>Design of opti</b>	<b>4</b>
<b>5</b>	<b>Program Structure</b>	<b>5</b>
5.1	Class Design . . . . .	5
5.2	Sequence Design . . . . .	6
5.3	Activity Design . . . . .	6
<b>6</b>	<b>Testing Cases</b>	<b>7</b>
<b>7</b>	<b>Performance analysis</b>	<b>8</b>
<b>8</b>	<b>Program Setup &amp; Analysis</b>	<b>9</b>
8.1	Program Setup . . . . .	9
8.2	Progarm Analysis . . . . .	9
<b>9</b>	<b>Appendix</b>	<b>10</b>

# 1 Introduction

The project aims to utilize the knowledge covered in COMP2432 Operating Systems and put them into practice to get a further understanding and improvement. This works out by implementing a facility management system for a company, which would make advantage of all kinds of scheduling skills and abstractions covered in the lecture, along with the use of multi-process programming and inter-process communication.

## 2 Scope

Multi-Process Programming and Inter-process Communication: `pipe()` and `fork()`;

CPU Scheduling: FCFS and PR

Memory Allocation: MFT

Synchronization: Program-Monitored Synchronization

## **3 Concept**

### **3.1 FCFS Scheduling**

FCFS scheduling is indeed taking the arriving time of the request as the priority. It is very easy to implement this algorithm, as long as the order of the request array is the same order as input, which is very natural. One thing to note that to keep the order of request array the same as input order, Input Module must be single-threaded.

### **3.2 PRIO Scheduling**

It is already mentioned that FCFS is a special case of PRIO scheduling where the priority is the arriving time. Now that a priority is appended with the request, all it needs is to sort the request array by priority, then call FCFS scheduling to implement PRIO scheduling. This reuses the code and decrease the complexity of the program.

## 4 Design of opti

## 5 Program Structure

### 5.1 Class Design

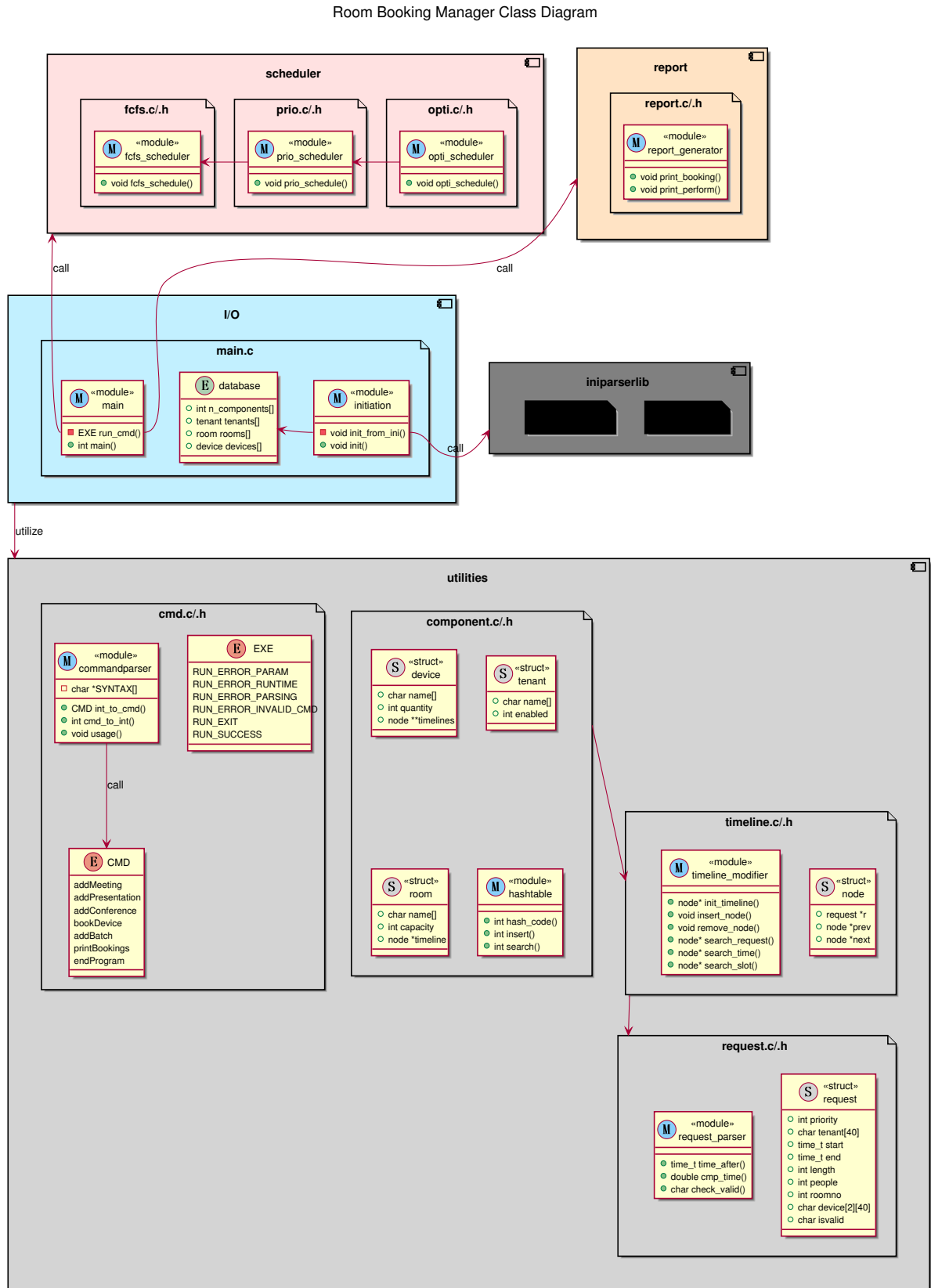


Figure 1: Overall class design diagram of Room Booking Manager

## 5.2 Sequence Design

Room Booking Manager Sequence Diagram

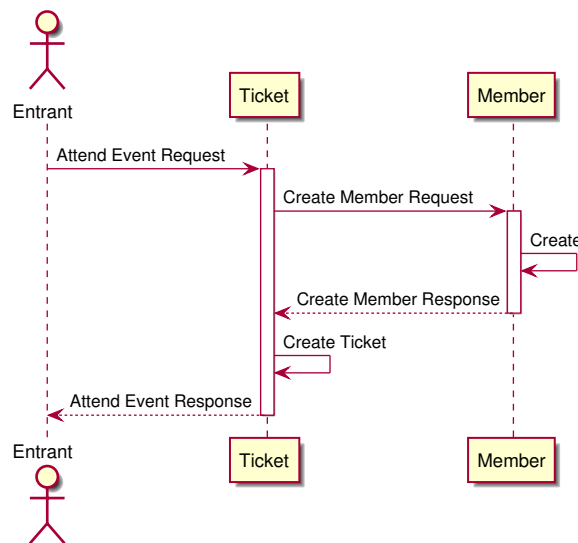
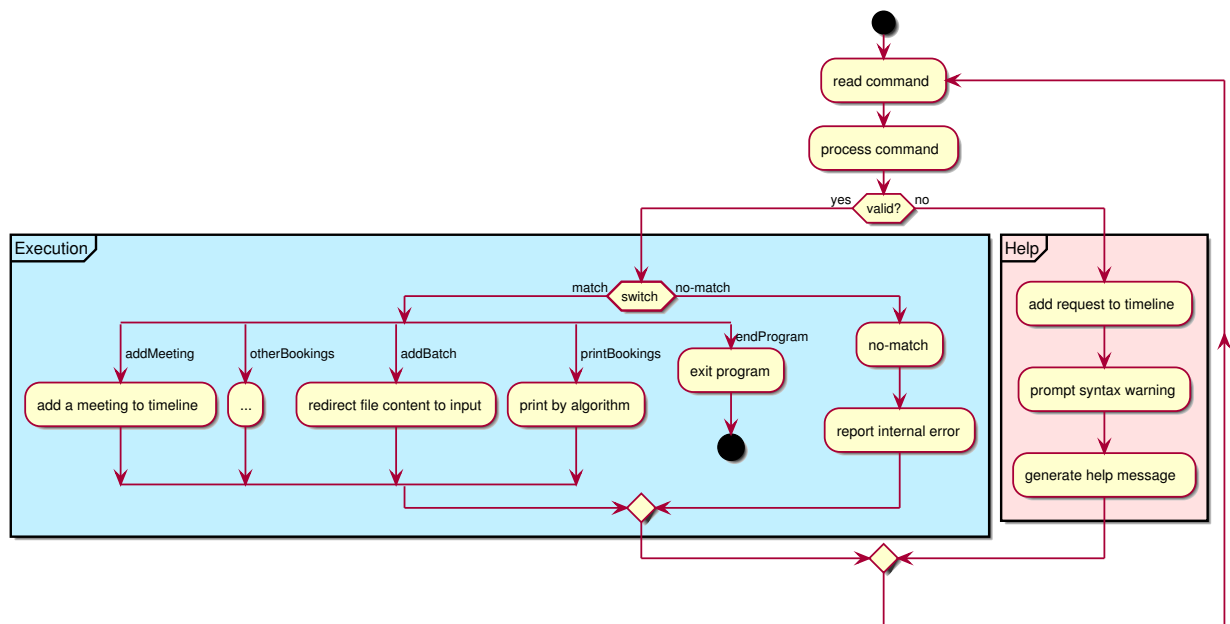


Figure 2: Overall sequence design diagram of Room Booking Manager

## 5.3 Activity Design

Room Booking Manager Activity Diagram





## 6 Testing Cases

This is the brief version.

Below demonstrates the valid and invalid tests for addMeeting instruction.

Tests for other instructions are similar but vary in parameters.

Refer to appendix for full version.

### valid tests

```
// the valid syntax is:
// addMeeting -tenant YYYY-MM-DD hh:mm n.n [d1 d2];
addMeeting -tenant_A 2021-05-10 21:50 1.50 5 projector_2K screen_100;
addMeeting -tenant_A 2021-05-11 18:20 0.30 5;
addMeeting -tenant_A 2021-05-11 4:10 0.0 5;
```

### invalid tests

```
// invalid instructions are either:
// command is invalid
command_invalid and parameters does not matter;
// tenant invalid
addMeeting -tenant_invalid 2021-05-10 1:30 0.50 5 projector_2K screen_100;
// date invalid
addMeeting -tenant_A date-in-valid 1:30 0.50 5 projector_2K screen_100;
// hhmm invalid
addMeeting -tenant_B 2021-05-10 hhmm:invalid 1.50 5 webcam_FHD monitor_75;
// duration invalid
addMeeting -tenant_C 2021-05-10 18:30 duration.invalid 5 webcam_FHD monitor_50;
// people invalid
addMeeting -tenant_D 2021-05-10 10:40 0.0 peopleinvalid projector_2K screen_150;
// device invalid
addMeeting -tenant_D 2021-05-16 3:10 1.10 5 device_invalid monitor_50;
// device pairing invalid
addMeeting -tenant_E 2021-05-10 22:20 0.50 5 projector_4K monitor_50;
// device without pairing
addMeeting -tenant_E 2021-05-10 22:20 0.50 5 projector_4K;
```

## 7 Performance analysis

## 8 Program Setup & Analysis

### 8.1 Program Setup

#### Step 0 Clone repo(optional)

Clone the repo from Github if there is no local copy.

```
git clone https://github.com/toolsmax0/COMP2432_RBM.git
```

#### Step 1 Compilation

cd to the project's root directory and execute `build.sh` script.

The program have dependency upon `gcc 4.0+` and `linux 3.0+`.

```
cd COMP2432_RBM
sh build.sh
```

#### Step 2 Costomization(optional)

To modify the component settings (i.e. tenants, rooms, devices), modify `RBM.ini` file according to its syntax.

#### Step 3 Execution

To execute the program, run the following command.

```
./out/RBM
```

### 8.2 Progarm Analysis

## 9 Appendix