Hostel Management System

Software requirements specifications

Version 1.0

Group 6 Members

Yan Paing Oo 3CS-93

Ye Thu Aung 3CS -132

La Min Thant 3CS -134

Thet Eain San 3CS – 116

Aye Myat Mon 3CS – 107

Shoon Lei Phyu 3CS - 102

Table Of Contents

TITLE PAGE NUMBER

1. Introductions………………………………………………… 2

1.1 Purpose ……………………………………………… 2

1.2 Scope Of the Project ………………………………… 2

1.3 Glossary ……………………………………………... 2

1.4 References …………………………………………… 3

1.5 Overview of Documents ……………………………... 3

2. Use Case Diagram ………………………………………….. 4

3. Use Case Specification ……………………………………… 5

3.1 Use Case Specification for Student Register ……….. 5

3.2 Use Case Specification for Generate Room ID …….. 6

3.3 Use Case Specification for Login …………………... 7

3.4 Use Case Specification for Check Rooms ………….. 8

3.5 Use Case Specification for Calculate Bills …………. 9

3.6 Use Case Specification for Pay Bills ……………… 10

3.7 Use Case Specification for Fine Bills……………….. 11

3.8 Use Case Specification for Notify Bills ……………… 12

4. Functional Requirements ……………………………………. 13

5. Non – Functional Requirements …………………………….. 14

6.0 ER Diagram ……………………………………………… 15

7.0 Class Diagram …….….…………………………………… 16

8.0 Data Dictionary …………………………………………….. 17

9.0 Data Definition Language …………………………………... 20

# 1.0.Introduction

# 1.1.Purpose

The purpose of this document is to manage various activities in hostel. It deals with the stuffs managed manually and avoid human mistakes. It can less human effort and time consuming. This document also provides data.

# 1.2.Scope of Project

This system goal is to design a robust software for hostel. This system will be a good design for students who want to stay in hostel. This system allows students to register and login to get rooms. Admin will also notify to pay monthly fees to student.

|  |  |
| --- | --- |
| Admin | Person who generate room IDs to students who want to stay in hostel. He can also login to check whether the rooms are available or not. If the rooms is free, he will give room id to student. He will also maintain bill. He will notify the students to pay monthly fees. He will also notify and get fine from students if the students do not give monthly bills. |
| Database | Collection of all the information about admin, students, rooms, hostel and fees. |
| Hostel | Hostel has many rooms and only 1 student can stay in 1 room. |
| Rooms | Rooms has their own room IDs. Admin can record the date when the student start to stay in and leave the room. Rooms has their room status whether the rooms are available or not. Rooms also have different categories and the amount of each room is also different. |
| Students | Person who stay in hostel. They have to login and do register with student id to get rooms from hostel. They will receive the rooms id from admin. They have to pay bill for the rooms. They will get notification from the admin if they do not pay monthly fees for the rooms. They will also have to give fines to the admin. This system goal is to design a robust software for hostel. This system will be a good design for students who want to stay in hostel. This system allows students to register and login to get rooms. Admin will also notify to pay monthly fees to student. |

# 1.3.Glossary

# 1.4.References

IEEE.IEE Std 830-1998 IEEE Recommended Practice for Software Requirements

Specifications. IEEE Computer Society.1998.

# 1.5.Overview of Document

The next chapter, the Overall Description section, of this document gives overview of the functionality of the hostel. It describes the informal requirements and details of the document.

The third chapter, the Overall Description section, of this document is written primarily for the admin and students and describes in technical terms the details of the functionality of the hostel.

The last chapter, the Overall Description section, of this document is written about non-functional requirements of the hostel ad describes the advantages of the system.

# 2.0 Use Case Diagram



# 3.0 USE CASE SPECIFICATIONS

# 3.1USE CASE SPECIFICATIONS FOR STUDENT REGISTER

|  |  |  |  |
| --- | --- | --- | --- |
| Number | 1 | | |
| Name | Registration | | |
| Summary | A student (user) needs to register to reserve a room | | |
| Priority | 1 | | |
| Preconditions | User opens the system | | |
| Post conditions | User will get the room ID | | |
| Primary Actor(s) | Student (user) | | |
| Secondary Actor(s) | Database of the system | | |
| Trigger | A student register to book a room | | |
| Main | | Step | Action |
| 1  2 | The system will let the user to fill the register form  User must fill the registration form and submit it |
|  | |

# 3.2 USE CASE Specifications For Generate Room ID

|  |  |  |  |
| --- | --- | --- | --- |
| Number | 2 | | |
| Name | Generating rooms ID | | |
| Summary | System manager (admin) will generate room ID for students who have registered | | |
| Priority | 1 | | |
| Preconditions | Admin will check the room availability | | |
| Post conditions | Admin will generate the room ID | | |
| Primary Actor(s) | System manager (admin) | | |
| Trigger | Admin will check available rooms and generate room ID | | |
| Main | | Step | Action |
| 1  2 | The system will show the admin vacant room  Admin will check and give the room ID to registered students |
| Open Issues | |  |  |
|  | |

# 3.3 USE CASE SPECIFICATIONS FOR LOGIN

|  |  |  |  |
| --- | --- | --- | --- |
| Number | 3 | | |
| Name | Log in | | |
| Summary | A student (user) can login with generated room ID | | |
| Priority | 2 | | |
| Preconditions | User has to type the room ID | | |
| Post conditions | User will login to system | | |
| Primary Actor(s) | Student (user), Admin | | |
| Secondary Actor(s) | Database of the system | | |
| Trigger | User needs to login to pay bill via system | | |
| Main | | Step | Action |
| 1  2 | Student gets the given room ID  Student must fill that ID no. to the login form |
| Open Issues | |  |  |
|  | |

# 3.4 USE CASE SPECIFICATIONS FOR CHECK ROOMS

|  |  |  |
| --- | --- | --- |
| Number | 4 | |
| Name | Check Rooms | |
| Summary | Checking Available For Students | |
| Priority | 3 | |
| Preconditions | Admin turns on the alert. | |
| Post Conditions | Student is available or not in time. | |
| Actors | Admin, Student | |
| Main Scenario | Step | Action |
|  | 1 | Student step out of their respective rooms when alert is reminded. |
|  | 2 | Admin manually checks the rooms one after another. |
| Extensions | Step | Branching |
|  | 1a | If student is absent, admin records the student name, room number. |
| Open Issues |  | |

# 3.5 USE CASE SPECIFICATIONS FOR CALCULATE BILLS

|  |  |
| --- | --- |
| Number | 5 |
| Name | Calculate bill |
| Summary | Calculate bill payment |
| Priority | 4 |
| Preconditions | The admin calculates student’s bill. |
| Postconditions | The admin had calculated student’s bill and got the bill detailed. |
| Actors | Administrator |
| Main Scenario | 1. Admin checked student’s bill . 2. Admin calculated bill payment and checked. 3. After got calculation for student’s bill, ask for bill fees. 4. The student pays the bill. |
| Extensions |  |
| Open issue |  |

# 3.6 USE CASE SPECIFICATIONS FOR PAY BILL

|  |  |
| --- | --- |
| Number | 6 |
| Name | Pay bill |
| Summary | Pay bill payment |
| Priority | 3 |
| Preconditions | The student wants to pay the bill of hostel room. |
| Postconditions | The student had paid the bill successfully. |
| Actors | Student |
| Main Scenario | 1.The student for bill payment.  2.The admin search the student and open the student profile.  3. The admin enters the fee bill payment detail.  4.The student pays the bill. |
| Extensions |  |
| Open issue |  |

|  |  |  |
| --- | --- | --- |
| Number | 7 | |
| Name | Fine Bills | |
| Summary | Paying Fine Bills | |
| Priority | 3 | |
| Preconditions | Admin login to the system and checks Fees data. | |
| Post Conditions | Admin checks student payment due date. | |
| Actors | Admin, Student | |
| Main Scenario | Step | Action |
|  | 1 | Admin checks student’s monthly payment is paid or not. |
|  | 2 | If not , admin send notification to student that due date is over. |
|  | 3 | Also adds fine bills to student’s payment. |
| Extensions | Step | Branching |
|  | 1a | Admin update student’s payment into database. |
| Open Issues |  | |

# 3.7 USE CASE SPECIFICATIONS FOR Fine Bill

# 3.8 USE CASE Specifications FOR NOTIFY BILLS

|  |  |  |
| --- | --- | --- |
| Number | 8 | |
| Name | Notify for Bills | |
| Summary | Sending Notifications to Students | |
| Priority | 3 | |
| Preconditions | Admin logins to the system and checks notifications from students. | |
| Post Conditions | Admin checks student’s payment. | |
| Actors | Admin, Student | |
| Main Scenario | Step | Action |
|  | 1 | Admin update student’s payment. |
|  | 2 | Admin sends notifications to student with voucher for paid bills. |
| Extensions | Step | Branching |
| Open Issues |  | |

# 4.0 Functional Requirements

Functional requirements The major functional requirements can be listed as follows

⁃ Allow a user to register as a member in the system

⁃ After registration, the user will get room id information and system will show which room is available or not to user .

⁃ When u get room id information, u can log in to the system to check your data

⁃ Can check which room can get or not

⁃ According to the registration, the system will calculate the amount of bill

⁃ The user must pay bill depending on the amount of calculation bill

⁃ The person can check how much bill he requires to give cos of his due date bill conditions

⁃ The person will get information of pay bill and also fine bill

# 5.0 Non-Functional requirements

Non- functional requirements are also equally important as functional requirements because it will affect the equality of the target system.  
  
1. **Security:**

The software used for this hostel management system include the password, so the security is provided. When anyone opens the software it has the provision for entering password. We have to enter the correct password; otherwise we cannot enter into the system. Password is saved in system registry for more security.  
  
2. **User Friendly**:

This package is very user friendly because it is easy to maintain and operate. All data entry operations are simple, administrator wants only enter data and all other operations are performed by the computer.  
  
3. **Availability:**  
System should be able to work for all 24 hours of a day i.e all the time system is available for security purpose.  
  
4. **Portability:**  
Our system is not depended on any platform, It can run on any OS.

5.**Performance:**

The system shall support up to 2 students per room.

6**.Maintainability:**

The system shall provide the capability to backup the database.

7**.Reliability:**

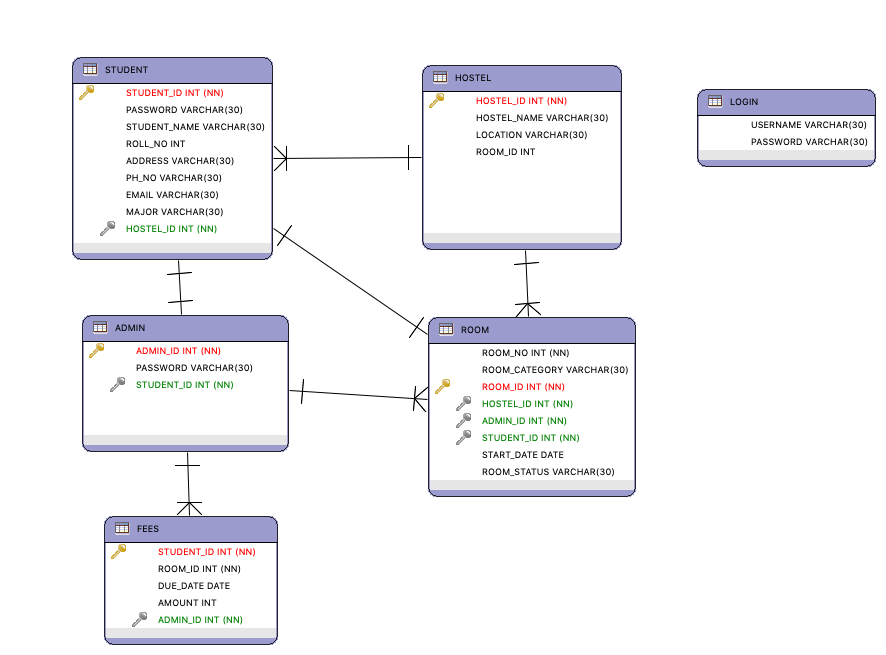
The system shall be available 99.9% of the time.

8**.Other constrains:**

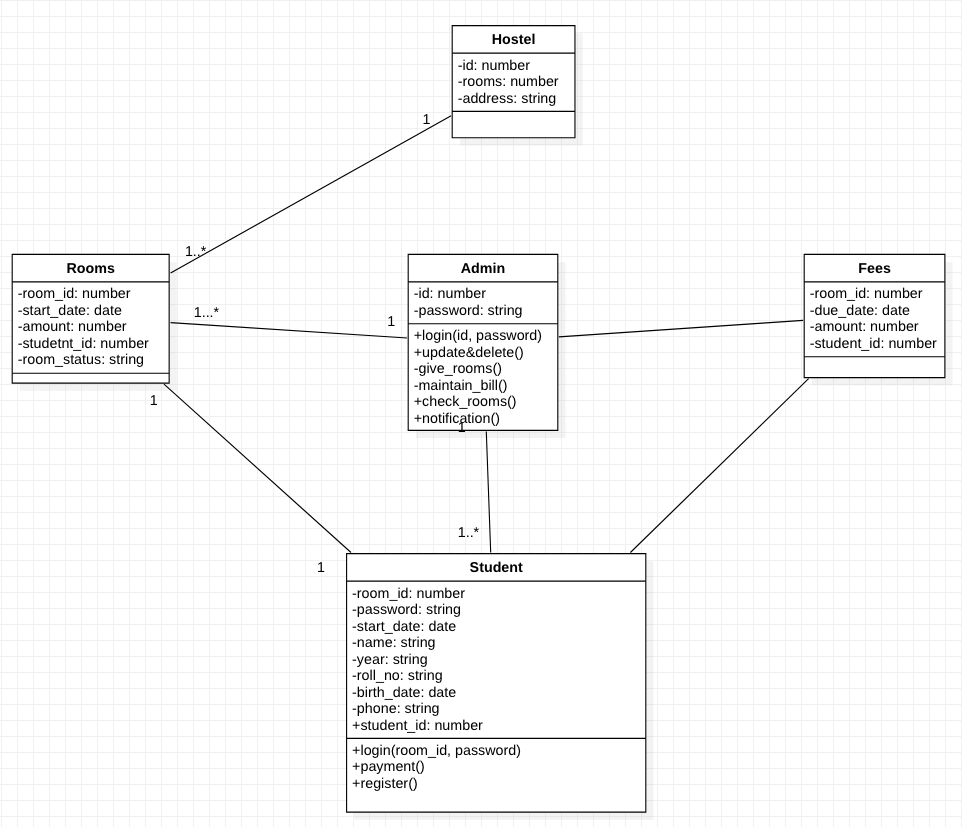
The system shall support barcode scanning of ID cards and hostel issues.

The HMS shall be flexible and adaptable due to future plans of expanding the system.

6.0 ER DIAGRAM



7.0 Class Diagram



8.0 Data Dictionary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table | Column | Data Type | PK | FK | Not Null |
| Admin | Admin ID | Int (11) | Y | N | N |
| Password | Varchar (30) | N | N | N |
| Student ID | Int (11) | N | Y | N |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table | Column | Data Type | PK | FK | NOT NULL |
| Student | Student ID | Int(11) | Y | N | N |
| Room ID | Int(11) | N | N | N |
| Due Date | Date | N | N | Y |
| Amount | Int(11) | N | N | Y |
| Admin ID | Int(11) | N | Y | N |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table | Column | Data Type | PK | FK | NOT NULL |
| Hostel | Hostel ID | Int(11) | Y | N | N |
| Hostel Name | Varchar(30) | N | N | N |
| Location | Varchar(30) | N | N | N |
| Rooms | Int(11) | N | N | N |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table | Column | Data Type | PK | FK | NOT NULL |
| Login | Username | Varchar(30) | N | N | N |
| Password | Varchar(30) | N | N | N |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table | Column | Data Type | PK | FK | NOT NULL |
| Room | Room No | Int(11) | N | N | N |
| Room Category | Varchar(30) | N | N | Y |
| Room ID | Int(11) | Y | N | N |
| Hostel ID | Int(11) | N | Y | N |
| Admin ID | Int(11) | N | Y | N |
| Student ID | Int(11) | N | Y | N |
| Start Date | Date | N | N | N |
| Room Status | Varchar(30) | N | N | N |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table | Column | Data Type | PK | FK | NOT NULL |
| Student | Student ID | Int(11) | Y | N | N |
| Password | Varchar(11) | N | N | N |
| Student Name | Varchar(11) | N | N | N |
| Roll No | Int(11) | N | N | Y |
| Address | Varchar(30) | N | N | Y |
| Ph No | Varchar(30) | N | N | Y |
| Email | Varchar(30) | N | N | Y |
| Major | Varchar(30) | N | N | Y |
| Room ID | Int(11) | N | Y | N |

9.0 Data definition language

SET SESSION FOREIGN\_KEY\_CHECKS=0;

/\* Drop Tables \*/

DROP TABLE IF EXISTS FEES;

DROP TABLE IF EXISTS ROOM;

DROP TABLE IF EXISTS ADMIN;

DROP TABLE IF EXISTS STUDENT;

DROP TABLE IF EXISTS HOSTEL;

DROP TABLE IF EXISTS LOGIN;

/\* Create Tables \*/

CREATE TABLE ADMIN

(

ADMIN\_ID INT NOT NULL,

PASSWORD VARCHAR(30),

STUDENT\_ID INT NOT NULL,

PRIMARY KEY (ADMIN\_ID)

);

CREATE TABLE FEES

(

STUDENT\_ID INT NOT NULL,

ROOM\_ID INT NOT NULL,

DUE\_DATE DATE,

AMOUNT INT,

ADMIN\_ID INT NOT NULL,

PRIMARY KEY (STUDENT\_ID)

);

CREATE TABLE HOSTEL

(

HOSTEL\_ID INT NOT NULL,

HOSTEL\_NAME VARCHAR(30),

LOCATION VARCHAR(30),

ROOMS INT,

PRIMARY KEY (HOSTEL\_ID)

);

CREATE TABLE LOGIN

(

USERNAME VARCHAR(30),

PASSWORD VARCHAR(30)

);

CREATE TABLE ROOM

(

ROOM\_NO INT NOT NULL,

ROOM\_CATEGORY VARCHAR(30),

ROOM\_ID INT NOT NULL,

HOSTEL\_ID INT NOT NULL,

ADMIN\_ID INT NOT NULL,

STUDENT\_ID INT NOT NULL,

START\_DATE DATE,

ROOM\_STATUS VARCHAR(30),

PRIMARY KEY (ROOM\_ID)

);

CREATE TABLE STUDENT

(

STUDENT\_ID INT NOT NULL,

PASSWORD VARCHAR(30),

STUDENT\_NAME VARCHAR(30),

ROLL\_NO INT,

ADDRESS VARCHAR(30),

PH\_NO VARCHAR(30),

EMAIL VARCHAR(30),

MAJOR VARCHAR(30),

ROOM\_ID INT NOT NULL,

PRIMARY KEY (STUDENT\_ID)

);

/\* Create Foreign Keys \*/

ALTER TABLE FEES

ADD FOREIGN KEY (ADMIN\_ID)

REFERENCES ADMIN (ADMIN\_ID)

ON UPDATE RESTRICT

ON DELETE RESTRICT

;

ALTER TABLE ROOM

ADD FOREIGN KEY (ADMIN\_ID)

REFERENCES ADMIN (ADMIN\_ID)

ON UPDATE RESTRICT

ON DELETE RESTRICT

;

ALTER TABLE ROOM

ADD FOREIGN KEY (HOSTEL\_ID)

REFERENCES HOSTEL (HOSTEL\_ID)

ON UPDATE RESTRICT

ON DELETE RESTRICT

;

ALTER TABLE STUDENT

ADD FOREIGN KEY (ROOM\_ID)

REFERENCES ROOM (ROOM\_ID)

ON UPDATE RESTRICT

ON DELETE RESTRICT

;

ALTER TABLE ADMIN

ADD FOREIGN KEY (STUDENT\_ID)

REFERENCES STUDENT (STUDENT\_ID)

ON UPDATE RESTRICT

ON DELETE RESTRICT

;

ALTER TABLE ROOM

ADD FOREIGN KEY (STUDENT\_ID)

REFERENCES STUDENT (STUDENT\_ID)

ON UPDATE RESTRICT

ON DELETE RESTRICT

;