Software Requirements Specification

for

Hostel Management System

Version 1.0

Prepared by

Group Number: 4

Emmanuel Marian Mathew	**B180347CS**	**emmanuel_b180347cs@nitc.ac.in**
Mathew Jose Mammoottil	**B180586CS**	**mathew_b180586cs@nitc.ac.in**
Shaaheen A M	**B181134CS**	**shaaheen_b181134cs@nitc.ac.in**
Sidharth Menon	**B180561CS**	**sidharth_b180561cs@nitc.ac.in**
Vishnu Sajith	**B180474CS**	**vishnu_b180474cs@nitc.ac.in**

Instructor: Abdul Nazeer

Course: Database Management System

Date: 19 - 10 - 2020

CONTENTS I

1 Introduction

<TO DO: Please provide a brief introduction to your project and a brief overview of what the reader will find in this section.>

1.1 Document Purpose

The product chosen for this project is a website or a mobile application to facilitate a hostel management system. The software package is to be developed from scratch, exclusively for the National Institute of Technology, Calicut and aims to,

- Optimize efficiency while managing a vast database
- Help save time and human effort
- Maintain and process complete details of all Students staying in the hotels
- Minimize human error affecting the database
- Implement high security
- Improve data consistency

1.2 Product Scope

The scope of this Hostel Management System software database includes but is not limited to:

- Providing a friendly interface for the users and the admin
- Allowing the easy search for *occupants* and their details
- Minimizing human error while managing the database
- Avoiding data redundancies in the database
- Ensuring high security of the data
- Facilitating the easy access to all the details of any student for the administration
- Recognizing *complaints* or *requests* for maintenance by the *occupants*
- Listing of *Lost and Found* items in the hostel
- Potentially developing the Database Management System for full-time use to maintain the database of the NIT Calicut hostels.

1.3 Intended Audience and Document Overview

This document is intended for the instructors of the course C3002D 'Database Management Systems' for their review and monitoring progress of the project. The document is also for the project team to use in order to analyze the system requirements and any additional details related to the project.

1.4 Definitions, Acronyms and Abbreviations

- HTML - HyperText Markup Language
- CSS - Cascading Style Sheets
- IEEE - Institution of Electrical and Electronic Engineers
- SRS - Software Requirement Specification
- HMS - Hostel Management System

1.5 Document Conventions

The document conventions used in the preparation of this SRS for the hostel management system are listed below:

- All key-words related to Hostel Management are formatted in italics.
- Page numbers are mentioned on the top right corner of every page.
- All general description of the project is typed in the Arial font.

1.6 References and Acknowledgments

The following references were used in creating this SRS document.

- The sample SRS template received from the course faculty.
- https://cs.gmu.edu/~rpettit/files/project/SRS-template.doc
- http://www.cse.msu.edu/~cse870/IEEEXplore-SRS-template.pdf

2 Overall Description

2.1 Product Overview

The Hostel Management System is a web, or a mobile application developed to let the hostel authorities maintain a database of the *occupants* of the hostel, share information and keep track of the *occupants* or students. It also lets the students or the *occupants* to lodge *complaints*, add

suggestions, and access the latest updates from the authorities. The application allows the user to sign up as an administrator or as a student. After signing up, each time they use the application they can login and continue to use the application in their preference. Depending on the user, the application will have a different user interface. For instance, if the user is an administrator, he/she will have certain features to allocate rooms to *occupants*, search for *occupants*, add *visitor* details, view suggestions and *complaints* and so on. If the user is a student, he/she will have features like check *mess dues*, search for *lost and found*, complaint or request for maintenance etc.

The main aim of implementing the Hostel Management System is to reduce human error, strength, and strain of manual labour, implement high security, avoid data redundancy, improve data consistency and to provide a platform to get things done in our fingertips.

2.2 Product Functionality

The following functions are planned to be implemented in the application:

1. The application must be able to authenticate the login id and be able to create a new login for the students residing in the hostels and for the staff and administrator as well.

2. Using the application, the administrator must be able to:
    a. Allot students to various rooms
    b. Search for *occupants*.
    c. Display details of the *occupants*.
    d. Modify the details of the *occupants*.
    e. Filter out *occupants* based on various parameters like:
        1. *Mess dues*.
        2. Fines.
        3. Location.
        4. Year of study etc.
    f. Allocate students to different *messes.*
    g. Maintain and monitor *mess dues* of students.
    h. Issue the due-list.
    i. Update the due-list.
    j. Add *visitor* details.

3. It must be able to let the *occupants* initiate *requests* and *complaints* about maintenance, cleaning, *mess* and so on.

4. The students must be able to post *lost and found* items and queries such as clarification of *mess dues* etc.

2.3 Design and Implementation Constraints

The development of the system will be constrained by the software used to make the application. The Flutter software used to make the mobile application will only be compatible for making android apps for users. For the web app we need a PC with 4 GB RAM. These are the hardware

and software constraints in the design and implementation of the web or mobile application of the Hostel Management System.

2.4 Assumptions and Dependencies

The following are the assumptions and dependencies that are imposed upon the implementation of the HMS:

1. The application must have a user-friendly interface that is simple enough for all types of users to understand.

2. The user of the application is expected to have a general knowledge of basic computer skills and the internet.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

For Mobile Application:

1. Flutter - App Development
2. Firebase - Database Management

For Web Application:

1. Front end - HTML, CSS, Bootstrap and Javascript
2. Back end - Python Flash
3. MYSQL - Database Management

3.1.2 Hardware Interfaces

For Mobile Application:

Android Versions from Version 4.1

For Web Application:

1. Windows Operating system
2. A browser which supports HTML, CSS, Bootstrap and Javascript

3.1.3 Software Interfaces

For Mobile Application:

Software used	**Description**
Operating System	We have chosen Android due to its immense popularity and rich application framework
Firebase	For Database Management
Flutter	For Application Development

For Web Application:

Software used	**Description**
Operating System	We have chosen Windows for its best support and user-friendliness.
Front end of web application	HTML, CSS, Bootstrap and Javascript

Back end of web application	Python Flask
Database	MYSQL

3.2 Functional Requirements

3.2.1 **F1:** Students should be able to enter their personal details for registration.

3.2.2 **F2:** Admin should be able to allocate rooms to the new students and add their details.

3.2.3 **F3:** Admin should be able to search for the *occupants*, access and display their details.

3.2.4 **F4:** Admin should be able to modify the occupant details.

3.2.5 **F5:** Admin should be able to filter out *occupants* based on various parameters.

3.2.6 **F6:** Admin should be able to delete the record of the *occupants* after *occupants* vacating.

3.2.7 **F7:** Students should be able to enter details of and search for *lost and found* items.

3.2.8 **F8:** Students should be able to add *visitor* details.

3.2.9 **F9:** Admin should be able to allocate students to different *messes*.

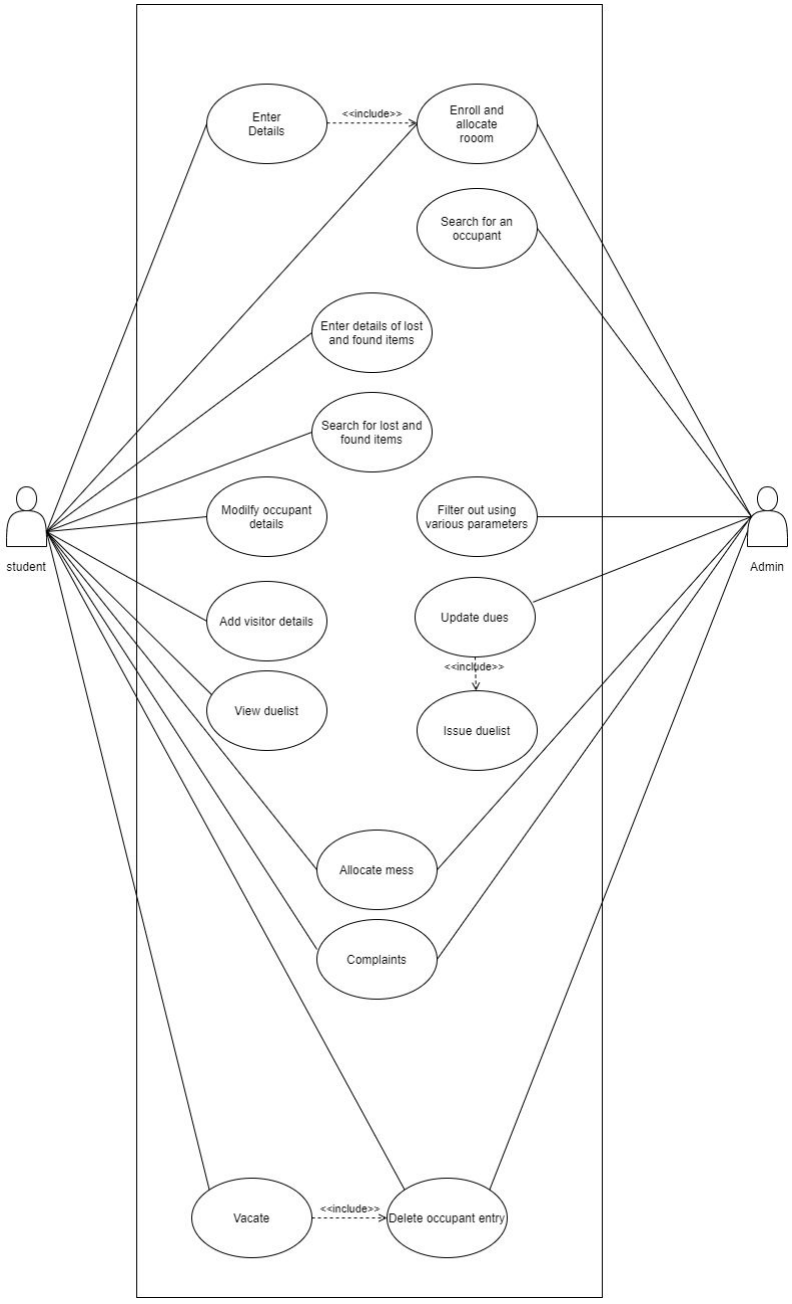
3.2.10 **F10:** Admin should be able to maintain and monitor occupant *dues*.

3.2.11 **F11:** Admin should be able to issue duelist.

3.2.12 **F12:** Students should be able to request for or raise *complaints* about room maintenance and appliances.

3.2.13 **F13:** Students should be able to view details of their *dues*.

3.3 Use Case Model



3.3.1 Use Case #1: Enter student Details (U1)

Purpose - Collect the details of new occupant
Priority - High
Actors - Student
Extends - NIL
Flow of Events -

1. Basic flow - Student enters their details like Name, Roll Number and Address one column at a time
2. Alternate flow - If the student enters invalid information type/ no information in any column, the system asks for input again
3. Exception - Student can abandon the process at any time

Includes - NIL

3.3.2 Use Case #2: Enroll the student and Allocate room (U2)

Purpose - To enter the new occupant details into the database and allocate room number
Priority - High
Actors - Student, Admin
Extends - NIL
Flow of Events -

1. Basic flow -
    a. Admin checks if the student entry already exists in the database
    b. If not, a new entry in the database is made with all the entered details
    c. A room is allocated to the student
2. Alternate flow -
3. Exception - All rooms are full, Entry already exists in the database.

Includes - U1

3.3.3 Use Case #3: Search for occupant (U3)

Purpose - Search for one particular occupant using roll number.
Priority - Medium
Actors - Admin
Extends - NIL
Flow of Events -

1. Basic flow -
    a. Admin enters the Roll Number of the occupant whose details are required
    b. The details of the occupant are displayed
2. Alternate flow - If the admin does not enter a valid Roll Number the system asks for input again
3. Exception - Entry corresponding the Roll Number does not exist

Includes - NIL

3.3.4 Use Case #4 - Enter Details of *Lost and Found* items (U4)

Purpose - Collecting details of lost/found items
Priority - Low
Actors - Student

Extends - NIL
Flow of Events -

1. Basic flow -
    a. Select the category of item (lost or found)
    b. Select the type of item (ID or purse etc.)
    c. Enter a brief description
    d. Enter phone number
2. Alternate flow -
    a. If no category or no type is selected, the user is asked to make the selection
    b. If description or phone number column is left empty, the user is asked to enter details
    c. If in the phone number column text is entered, the user is asked to provide a valid input.
3. Exception - Student can abandon the process at any time

Includes - NIL

3.3.5 Use Case #5 - Search for *Lost and Found* items (U5)

Purpose - Collecting details of lost/found items
Priority - Low
Actors - Student
Extends - NIL
Flow of Events -

1. Basic flow -
    a. Select the category of item (lost or found)
    b. Select the type of item (ID or purse etc.)
    c. Display the list of items under corresponding category and type
2. Alternate flow -
    If no category or no type is selected, the user is asked to make the selection
3. Exception - Student can abandon the process at any time, no item found in the search

Includes - NIL

3.3.6 Use Case #6 - Modify Occupant Details (U6)

Purpose - Modify occupant details of allowed/modifiable attributes
Priority - Medium
Actors - Student
Extends - NIL
Flow of Events -

1. Basic flow -
    a. Enter Roll Number
    b. Select the attribute to be modified (only modifiable attributes will be listed here)
    c. Enter the new information

2. Alternate flow -
    a. If Roll Number entered is of invalid format/ NIL, then the user is asked to perform input again
    b. If no category is selected, the user is asked to make the selection
    c. If the new information entered is of invalid type or empty or no change is made to the already existing information, the user will be asked to perform step c in basic flow again
3. Exception - Student can abandon the process at any time, Entry with entered roll number does not exist in the database

Includes - NIL

3.3.7 Use Case #7 - Add *Visitor* Details (U7)

Purpose - Add *visitor* details
Priority - Low
Actors - Student
Extends - NIL
Flow of Events -

1. Basic flow -
    a. Enter Roll Number
    b. Enter the *visitor* details like Name, Contact Details and Date of Visit
2. Alternate flow - If any field is left empty/ invalid the user will asked to enter valid input
3. Exception - Student can abandon the process at any time, Entry with entered roll number does not exist in the database

Includes - NIL

3.3.8 Use Case #8 - Filter Search (U8)

Purpose - Filter out *occupants* based on various parameters
Priority - Medium
Actors - Admin
Extends - NIL
Flow of Events -

1. Basic flow -
    a. Admin selects the filter category (Location, Year of Study)
    b. Admin selects the filter condition from all possible options under that category
    c. The list of *occupants* satisfying the filter conditions is displayed.
2. Alternate flow -
    a. If the admin does not select a category, they are asked to make a selection.
    b. If the admin does not select a condition, they are asked to make a selection.
3. Exception - No entry under the category, No entry satisfying the condition

Includes - NIL

3.3.9 Use Case #9 - Update *dues* (U9)

Purpose - Monthly update the *dues* field in the database of all students
Priority - High
Pre Conditions - Document with new *mess dues* should be available and accessible
Actors - Admin
Extends - NIL
Flow of Events -

1. Basic flow -
    a. Take roll number each entry in the new *mess dues* document
    b. Update *dues* field of the corresponding entry in our database
2. Alternate flow -
    a. If the roll number is not present in the database, skip to the next entry in the document
3. Exception - Entry missing in the document

Includes - NIL

3.3.10 Use Case #10 - Issue Duelist (U10)

Purpose - Issue the document containing updates due details of all students
Priority - High
Actors - Admin
Extends - NIL
Flow of Events -

1. Basic flow -
    a. Select roll number and *dues* fields from the database
    b. Export the selection as a document which can be accessed by students.
2. Alternate flow -
3. Exception - Due field of an entry is empty

Includes - U9

3.3.11 Use Case #11 - View Duelist (U11)

Purpose - View the issued duelist
Priority - High
Actors - Student
Extends - NIL
Flow of Events -

1. Basic flow -
    a. Select the option to view the duelist
    b. Open the duelist document

2. Alternate flow -
3. Exception - Document does not exist

Includes - NIL

3.3.12 Use Case #12: Allocate *Mess* (U12)

Purpose - Admins allocate *mess* to the student
Priority - High
Actors - Student, Admin
Extends - NIL
Flow of Events -

1. Basic flow -
    a. Student makes a request to be enrolled in a *mess* by selecting from the available options
    b. Admin checks whether the requested *mess* is full.
    c. If not, the student is enrolled in the requested *mess*.
2. Alternate flow -
    a. If the student does not select any option, they are asked to make a selection
    b. If the requested *mess* is full, the student is requested to select another *mess*
3. Exception -
    a. The student can terminate the process at any time
    b. The *mess dues* are not paid
    c. All the *messes* are full

Includes - NIL

3.3.13 Use Case #13 - *Complaints* (U13)

Purpose - Register *complaints* about room maintenance
Priority - Low
Actors - Student, Admin
Extends - NIL
Flow of Events -

1. Basic flow -
    a. The student enters the roll number, room number and *mess* allocated.
    b. The student selects the category of the complaint (*Mess*, Room maintenance etc.).
    c. Student enters a description of their complaint
    d. Admins review the complaint
2. Alternate flow -
    If the details, selection or description is missing/invalid, the user is requested to enter it.
3. Exception -

Includes- NIL

3.3.14 Use Case #14: Vacate (U14)

Purpose - The student vacates from the hostel
Priority - High
Actors - Student
Extends - NIL
Flow of Events -

2. Basic flow -
    a. The student enters the roll number, room number and *mess* allocated.
    b. Select the option verifying that they are going to vacate
2. Alternate flow - If the details or selection is missing/invalid, the user is requested to do it.
3. Exception -
    a. There's no entry in database with Roll Number entered
    b. *Mess* fees are not paid
    c. The student can terminate the process at any time.

Includes- NIL

3.3.15 Use Case #15: Delete Occupant Entry (U15)

Purpose - To delete the vacated student's entries from the database
Priority - High
Actors - Student, Admin
Extends - NIL
Flow of Events -

1. Basic flow - Deletes the entry of the student who is vacating
2. Alternate flow -
3. Exception - Entry does not exist in the database

Includes- U14

4 Other Non-functional Requirements

4.1 Performance Requirements

- The database should be able to store information of a large collection of records and the changes made from one user should be updated for all users in real time.
- The system must be responsive and should have less delays.
- The application should be able to support multiple users at a time.

4.2 Safety and Security Requirements

- The database may take time to load or crash in case a lot of users are trying to access together. In such cases, the restoration of the database might take time.
- Due to some virus, the operating system might take time to boot up which would cause certain delays.
- Critical information should be encrypted and sent to be stored in the server.
- Certain tasks or access to certain functionalities should be authorized and only those members should be able to use those functions.
- Keep a log on who modified data and restrict communication between certain parts of the application.
- Must prevent the accidental access, modification and destruction of the database.

4.3 Software Quality Attributes

- Availability Requirement: The system should be available to the user 24/7. The changes in relation to the last date to pay bills or the postponing of an event should reach all students. Hence the availability of the system must be high.

- Efficiency Requirement: In case of a system repair, scheduled software update or a system crash, the system should be able to restore and recover quickly.

- Accuracy: It should be able to provide precise real time data about the various users. It should regularly update the *dues* lies and the Administration should be able to know whether a student needs to pay fees or a *mess* card can be issued directly.

- Hardware Constraints: The system requires to store the details of the administration staff and hence the database should have backup capabilities if the data integrity of the system is compromised.

- Privacy Requirements: No user who is not an Administrator can access the functionalities of the Administration and only those students who are part of the college should have access to a room and *Mess*
- *Requests*
- privileges.

Appendix A - Group Log

<Please include here all the minutes from your group meetings, your group activities, and any other relevant information that will assist in determining the effort put forth to produce this document>

Google Meet on **6th October 2020:**

- The group met on google meet to decide on which topic to select for the project.
- Shortlisted different possible and viable options for the topic.
- We decided to proceed with the Hostel Management System.
- After fixing the topic we decided to take a day to brainstorm and analyse what all problems we faced with the present Hostel Management System.

Google Meet on **7th October 2020:**

- All of us made a list of the shortcomings in the present Hostel Management System and the necessary measures on how we can better the situation if our product is deployed.
- Compiled all the inputs from the group and started working on the Abstract.
- We decided to proceed with a mobile application as well as a Web application.
- What all platforms and softwares are to be used for the implementation is finalised.
- Finished working on the Abstract.

Google meet on **12th October 2020:**

- The group met on google meet to discuss on how to write the SRS.
- Sample SRS documents of the IEEE format were referred and understood the basic idea on how to write an SRS.
- We divided the topics present in the SRS among ourselves to ensure equal contribution among the group members.

WhatsApp Group discussions on doubts regarding the SRS document.

Google Meet on **17th October 2020:**

- Compiled all the individual works in a google doc and went through each other's work and proposed suggestions and modifications to improve the SRS.
- Added all the missing requirements in the SRS.
- Finished working on the SRS.

Google Meet on **19th October 2020:**

- The group met on google meet and proofread the SRS before submitting.