**Hostel Management System**

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

**Bachelor of Technology/Master of Technology**

In

**Computer Science and Engineering**

**School of Engineering and Sciences**

Submitted by

**GUDA VENKATA NARASIMHA SAI NAREN KOUSIK**

**(AP22110011364)**

**A picture containing text

Description automatically generated**

Under the Guidance of

**(Dr. Subhankar Ghatak)**

**SRM University–AP**

**Neerukonda, Mangalagiri, Guntur**

**Andhra Pradesh – 522 240**

**[April, 2024]**

# Certificate

Date: 20-Apr-2024

This is to certify that the work present in this Project entitled “**Hostel Management System**” has been carried out by **GUDA VENKATA NARASIMHA SAI NAREN KOUSIK** under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in **School of Engineering and Sciences**.

**Supervisor**

(Signature)

Prof. / Dr. [Name]

Designation,

Affiliation.

**Co-supervisor**

(Signature)

Prof. / Dr. [Name]

Designation,

Affiliation.

# Table of Contents

[Certificate i](#_gjdgxs)

[Acknowledgements iii](#_30j0zll)

[Table of Contents v](#_1fob9te)

[Abstract vii](#_3znysh7)

[List of Publications ix](#_2et92p0)

[Statement of Contributions xi](#_tyjcwt)

[Abbreviations xv](#_3dy6vkm)

[List of Tables xvii](#_1t3h5sf)

[List of Figures xix](#_4d34og8)

[List of Equations xxi](#_2s8eyo1)

[1.](#_17dp8vu) 1

[1.1](#_3rdcrjn) **Error! Bookmark not defined.**

[1.1.1](#_26in1rg) **Error! Bookmark not defined.**

[2.](#_lnxbz9) 2

[2.1](#_35nkun2) **Error! Bookmark not defined.**

[2.1.1](#_1ksv4uv) **Error! Bookmark not defined.**

[3.](#_44sinio) 3

[4.](#_2jxsxqh) 4

[5.](#_z337ya) 6

[References 11](#_3j2qqm3)

**Identification of Project related to DBMS (Hostel Management System):**

Hostel management is essential for accommodation management in schools, workplaces or other establishments that provide accommodation services. These systems manage many things such as person details, Guardian details, room allocation and price management. The use of a database management system (DBMS) is essential for efficient organization and access to hotel-related information.

**Project Background: Hostel Management System:**

Hostel Management System Project was launched in the areas of schools that provide accommodation to students. The school is renowned for its commitment to academic excellence and student well-being and is home to a variety of students from different backgrounds and backgrounds. As the number of students continues to increase, the management realizes that hotel management needs to be strengthened to improve the accommodation system, keep students happy, and improve management.

**Identification of Needs and Opportunities:**

The decision to implement a hostel management system stems from the need to address several key challenges and capitalize on opportunities for improvement. These include:

**Efficient Data Management**:

The proliferation of manual records and disparate systems makes it difficult to maintain accurate and up-to-date hostel-related data, leading to errors and inefficiencies in data management.

**Enhanced Student Experience:**

Improving the hostel experience for students is paramount. A modern hostel management system can streamline processes such as room allocation, complaint management, and communication, thereby enhancing student satisfaction and retention.

**Administrative Streamlining**:

The current manual processes result in administrative bottlenecks and delays. Automating tasks such as fee collection, room allocation, and maintenance scheduling can significantly reduce administrative burden and improve staff productivity.

**Data Security and Compliance**:

With the increasing importance of data security and privacy regulations, there is a need to ensure that hostel-related data is securely managed, accessed, and stored in compliance with regulatory requirements.

**Description of project:**

The Hostel Management System (HMS) project aims to create an efficient and user-friendly software solution for managing accommodation facilities in an educational institution. This system will automate various hostel management processes such as room allocation, fee collection, visitor registration, inventory management, and maintenance scheduling. By centralizing all hostel-related operations and data into one platform, the HMS will improve efficiency, enhance student satisfaction, and facilitate effective administration of hostel facilities.

**Key Features:**

**1. Room Allocation and Management:**

The system will facilitate the allocation and management of student accommodations based on their preferences, availability, and administrative rules. Hostel administrators can easily assign rooms, update occupancy status, and manage room transfers through the system's intuitive interface**.**

**2. Fee Collection and Billing:**

The HMS will automate fee collection processes, allowing students to view their hostel fees, make payments online, and receive electronic receipts. Hostel administrators can generate invoices, track payment statuses, and send reminders for overdue payments, ensuring timely and accurate fee management.

**3. Visitor Management:**

The system will enable hostel administrators to register and track visitors entering the hostel premises, ensuring security and accountability. Visitors will be required to provide identification details and obtain approval from residents or hostel staff before accessing the hostel facilities.

**4. Inventory and Asset Management:**

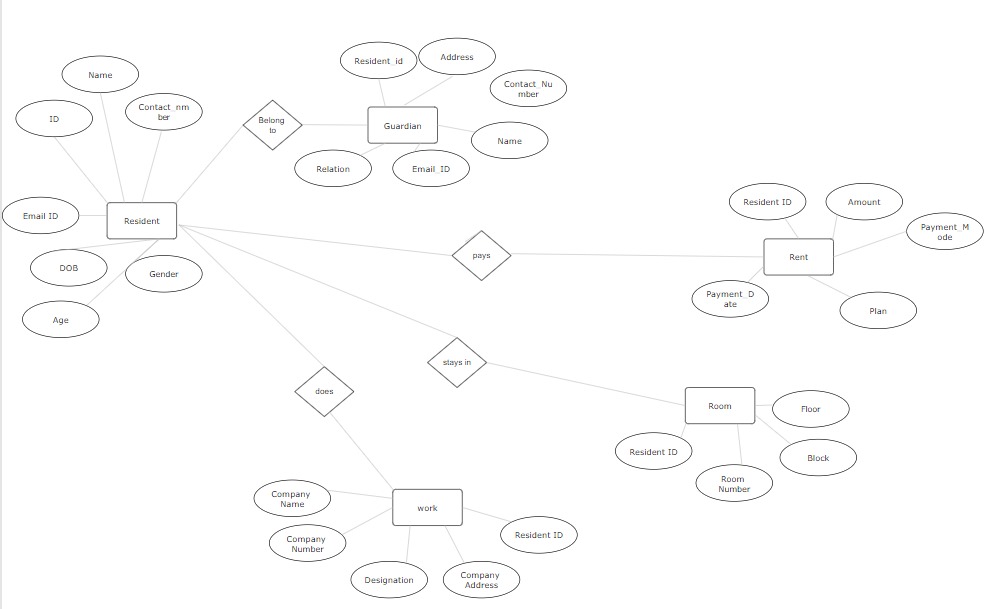
The HMS will maintain a comprehensive inventory of hostel assets, including furniture, appliances, and other amenities. Hostel administrators can track asset utilization, conduct inventory audits, and schedule maintenance tasks to ensure optimal functionality and resource allocation.

**5. Maintenance and Service Requests:**

Students can submit maintenance and service requests through the system, which will be automatically assigned to the relevant hostel staff. This feature will ensure timely resolution of issues and improve the overall maintenance and upkeep of hostel facilities.

In summary, the Hostel Management System will streamline and optimize the management of hostel facilities, providing a seamless experience for students and hostel administrators. By automating key processes and centralizing data, the system will improve efficiency, enhance student satisfaction, and facilitate effective administration of hostel facilities.

**ER Diagram Creation (use any online tools to draw ER diagram):**



**Description of ER diagram:**

->This ER diagram contains 5 entities

* Resident
* Guardian
* Rent
* Room
* Work

* Resident entity provides the information about person, who stays in the hostel
* Guardian entity provides the information about guardian details of the resident in the hostel
* Rent entity provides details about rent chosen by the resident
* Room entity provides room details of the resident
* Work entity provides the information about the place where resident does job or studies.
* Resident and guardian table are linked (Resident belongs to guardian)
* Resident and Rent table are linked (Resident pays rent)
* Resident and Room table are linked (Resident stays in the room)
* Resident and Work table are linked (Resident does work)

**Description of tables:**

**1.Resident table:**

This table contains details of person staying in hostel

* ID: The id number given to the resident to uniquely identify them (Auto\_incremented)
* Name: Name of the resident
* Date of Birth: Date of birth of the resident
* Age: Age of the resident (Derived from the Date of Birth)
* Gender: Sex. It should be male, female or others
* Contact\_Number: phone number of the resident
* Email\_ID: Email Id of the resident. It is optional

**2. Guardian table:**

This table contains details of guardian of person which is identified by Resident\_ID

* Resident\_ID: To identify the guardian to the resident (Foreign Key referred from Resident relation)
* Name: Name of the guardian
* Relation: Relation between guardian and the resident
* Contact\_Number: phone number of the guardian
* Email\_ID: Email Id of the guardian. It is optional
* Address: Address of the guardian

**3. Rent table:**

This table contains the rent details of person

* Resident\_ID: Id of the resident to identify his/her payment details (Foreign Key referred from Resident relation)
* Plan: Plan that can be chosen by the resident to pay. This may be annually, half-yearly, quarterly, monthly
* Amount: The money that the resident should pay according to his/her plan (Derived from the plan)
* Payment\_Date: Date that the resident had done his/her payment
* Payment\_Mode: Mode of the payment. Either Online or Offline

**4. Room table:**

This tables contain the room details of person that he chosen

* Resident\_ID: Id of the resident to identify his/her room details (Foreign Key referred from Resident relation)
* Room\_Number: room number, in which, the resident is staying in. It may be either 1 or 2 or 3 or 4 or 5
* Floor: floor of the room. Either First or Second or Third or Fourth
* Block: block of his/her stay. Either Block-1 or Block-2 or Block-3

**5. Work table:**

* Resident\_ID: Id of the resident to identify his/her work details (Foreign Key referred from Resident relation)
* Company\_Name: Name of the college or company, in which, the resident is doing his/her studies or job respectively
* Company\_Address: Address of the college or company
* Company\_Number: Contact number of college or company. It is optional
* Designation: Describes the resident is either student or employee

**Normalization of tables up to 3-NF:**

Normalization is a series of tests that individual relational schemas go through to ensure that the database is normalized to avoid redundancy as much as possible. It is a process of organizing data in database.

Normalization plays a crucial role in the designing of efficient(GOOD) database schema. It is in three forms

**1). First Normal Form (1NF):**

It is the test of the normalization process.

It ensures that the there are no multi-valued Attributes.

To Qualify this test, all columns should be atomic in nature(i.e.., No groups or arrays of attributes)

As we’ve seen before, there are no multi-valued attributes in the relations of our Database Schema. So it passes 1NF test.

**2). Second Normal Form (2NF):**

It is the second test in the normalization process. It ensures that the relation has no partial dependency.

Partial dependency is functional dependency condition in which, despite the removal an attribute from Y, the FD Y->X holds. In other words, it is a condition in which a non-prime attribute depends on some prime attributes only, instead of depending on all prime attributes.

In our database schema, as all the relations have exactly one prime attribute, It passed 2NF test too.

**3). Third Normal Form(3NF):**

It is the third test in the normalization process. It ensures that the relation has no transitive dependency.

Transitive Dependency is a functional dependency condition in which any two non-prime attributes involve in a functional dependency relationship.

In all the relations of the database schema, same kind of functional relationship can be observed

i.e.., Y->X where Y is a single attribute which is the ID (primary key) and X is the set of all other attributes.

So, this database schema passed 3NF test too.

With this, it marks the successful normalization process of the database schema.

**Creation of Data in the tables (at least 5 tables):**

**1). Resident table:**

INSERT INTO Resident (Name, DOB, Age, Gender, Contact\_Number, Email\_ID) VALUES (‘Resident\_1’, 01-01-2000, 24, ‘M’, 0123456789, ‘[Resident\_1@gmail.com](mailto:Resident_1@gmail.com)’);

INSERT INTO Resident (Name, DOB, Age, Gender, Contact\_Number, Email\_ID) VALUES (‘Resident\_2’, 01-01-2001, 23, ‘M’, 9876543210, ‘[Resident\_2@gmail.com](mailto:Resident_2@gmail.com)’);

**2). Guardian table:**

INSERT INTO Guardian (Name, Relation, Contact\_Number, Email\_ID, Address) VALUES (‘Guardian\_1’, ‘Father’, 1234567890, NULL, ‘abc’);

INSERT INTO Guardian (Name, Relation, Contact\_Number, Email\_ID, Address) VALUES (‘Guardian\_2’, ‘Mother’, 0987654321, NULL, ‘xyz’);

**3). Rent table:**

INSERT INTO Rent (Plan, Amount, Payment\_Date, Payment\_Mode) VALUES (‘Annually’, 100000, 20-04-2024, ‘Online’);

INSERT INTO Rent (Plan, Amount, Payment\_Date, Payment\_Mode) VALUES (‘Half-Yearly’, 50000, 20-04-2024, ‘Offline’);

**4). Room table:**

INSERT INTO Room (Room\_Number, Floor, Block) VALUES (1, ‘First Floor’, ‘Block-1’);

INSERT INTO Room (Room\_Number, Floor, Block) VALUES (2, ‘Second Floor’, ‘Block-2’);

**5). Work table:**

INSERT INTO Work (Company\_Name, Company\_Address, Company\_Number, Designation) VALUES (‘SRM University AP’, ‘Neerukonda’, NULL, ‘Student’);

INSERT INTO Work (Company\_Name, Company\_Address, Company\_Number, Designation) VALUES (‘SRM University AP’, ‘Neerukonda’, 0864322110, ‘Employee’);

**Write Sql queries (subqueries, aggregate functions, joins) on the created tables:**

**1). Insert operation (For joining new resident):**

1. **Resident table:**

INSERT INTO Resident (Name, DOB, Age, Gender, Contact\_Number, Email\_ID) VALUES ('value-1', 'value-2', value-3, ‘value-4’, value-5, 'value-6');

1. **Guardian table:**

SELECT MAX(ID) as Resident\_ID FROM Resident;

INSERT INTO Guardian (Resident\_ID, Name, Relation, Contact\_Number, Email\_ID, Address) VALUES (value-1, 'value-2', 'value-3', value-4, 'value-5', ‘value-6’);

1. **Rent table:**

SELECT MAX(ID) as Resident\_ID FROM Resident;

INSERT INTO Rent (Resident\_ID, Plan, Amount, Payment\_Date, Payment\_Mode) VALUES (value-1, 'value-2', value-3, 'value-4', 'value-5');

1. **Room table:**

SELECT MAX(ID) as Resident\_ID FROM Resident;

INSERT INTO Room (Resident\_ID, Room\_Number, Floor, Block) VALUES (value-1, value-2, 'value-3', 'value-4');

1. **Work table:**

SELECT MAX(ID) as Resident\_ID FROM Resident;

INSERT INTO Work (Resident\_ID, Company\_Name, Company\_Address, Company\_Number, Designation) VALUES (value-1, 'value-2', 'value-3', value-4, 'value-5');

**2). Update rent details (After expiry of current rent plan):**

UPDATE Rent SET Plan = ’value-6’, Amount = value-7, Payment\_Date = ’value-8’, Payment\_Mode= ’value-9’ WHERE Resident\_ID= value-1;

**3). Search using Select by joining views operation (Selecting by ID of the resident):**

SELECT t1. \*, t2. \*, t3. \*, t4. \*, t5. \*

FROM resident\_view t1

JOIN guardian\_view t2 ON t1. ID = t2. Resident\_ID

JOIN rent\_view t3 ON t1. ID = t3. Resident\_ID

JOIN room\_view t4 ON t1. ID = t4. Resident\_ID

JOIN work\_view t5 ON t1. ID = t5. Resident\_ID

Where t1. ID = Resident\_ID;

**4). Delete operation (For removing details of unnecessary residents):**

DELETE FROM Work WHERE Resident\_ID = value-1;

DELETE FROM Room WHERE Resident\_ID = value-1;

DELETE FROM Rent WHERE Resident\_ID = value-1;

DELETE FROM Guardian WHERE Resident\_ID = value-1;

DELETE FROM Resident WHERE ID = Resident\_ID;

**5). Joins used to fetch all details (For disposal of all resident details):**

SELECT t1. \*, t2. \*, t3. \*, t4. \*, t5. \*

FROM resident\_view t1

JOIN guardian\_view t2 ON t1. ID = t2. Resident\_ID

JOIN rent\_view t3 ON t1. ID = t3. Resident\_ID

JOIN room\_view t4 ON t1. ID = t4. Resident\_ID

JOIN work\_view t5 ON t1. ID = t5. Resident\_ID;

**Creation of views using the tables:**

**1). Resident view:**

CREATE VIEW resident\_view (ID, Name, DOB, Age, Gender, Contact\_Number, Email\_ID) AS

SELECT \* FROM Resident;

**2). Guardian view:**

CREATE VIEW guardian\_view (Resident\_ID, Guardian\_Name, Relation, Guardian\_Number, Guardian\_Email, Address) AS

SELECT \* FROM Guardian;

**3). Rent view:**

CREATE VIEW rent\_view (Resident\_ID, Plan, Amount, Payment\_Date, Payment\_Mode) AS

SELECT \* FROM Rent;

**4). Room view:**

CREATE VIEW room\_view (Resident\_ID, Room\_Number, Floor, Block) AS

SELECT \* FROM Room;

**5). Work view:**

CREATE VIEW work\_view (Resident\_ID, Company\_Name, Company\_Address, Company\_Number, Designation) AS

SELECT \* FROM Work;