DBMS Project Synopsis (Go + SQLite)

1. Title Page

Project Title: Hostel Management System

Course Name & Code: UCS310 – Database Management Systems

Degree & Year: B.Tech (2nd Year)

Department: Computer Science and Engineering

Institute Name: Thapar Institute of Engineering & Technology

Group Members:
Pranav Garg (Roll No: 1024170298)
Sachin Prakash (Roll No: 1024170302)

Lab Instructor: Dr. Tanya Garg

Academic Year: 2025–26

2. Introduction

The Hostel Management System is a database-driven web application designed to manage hostel operations efficiently. The system stores and manages student records, administrator details, and complaint tracking information.

Traditional manual systems suffer from data redundancy, inconsistency, and lack of security. A Database Management System (DBMS) provides structured data storage, integrity constraints, and efficient query processing.

This project emphasizes backend implementation using SQL and relational database concepts with SQLite as the database.

3. Problem Statement

In many hostels, student records and complaint tracking are handled manually or using spreadsheets. This leads to:

- Duplicate student records
- Poor complaint tracking
- No secure authentication system
- Difficulty in retrieving data
- Lack of data integrity

The proposed Hostel Management System provides a structured relational database solution to manage students and complaints efficiently.

4. Objectives of the Project

- To design a relational database for hostel management
- To implement primary and foreign key constraints
- To normalize the database up to Third Normal Form (3NF)
- To implement DDL and DML SQL commands
- To ensure referential integrity
- To manage complaint tracking effectively

5. Scope of the Project

Users:

- Admin
- Student

Modules:

- Admin Login Module
- Student Registration Module
- Complaint Submission Module
- Complaint Status Management Module

The system focuses primarily on backend database operations.

6. Proposed System Description

The system is developed using:

- Go (Golang) for backend logic
- SQLite for database
- HTML/CSS for frontend

Working of the System:

1. Admin logs in using credentials stored in the `admins` table.
2. Students register and their data is stored in the `students` table.
3. Students submit complaints.
4. Complaints are stored in the `complaints` table.
5. Admin can view and update complaint status.

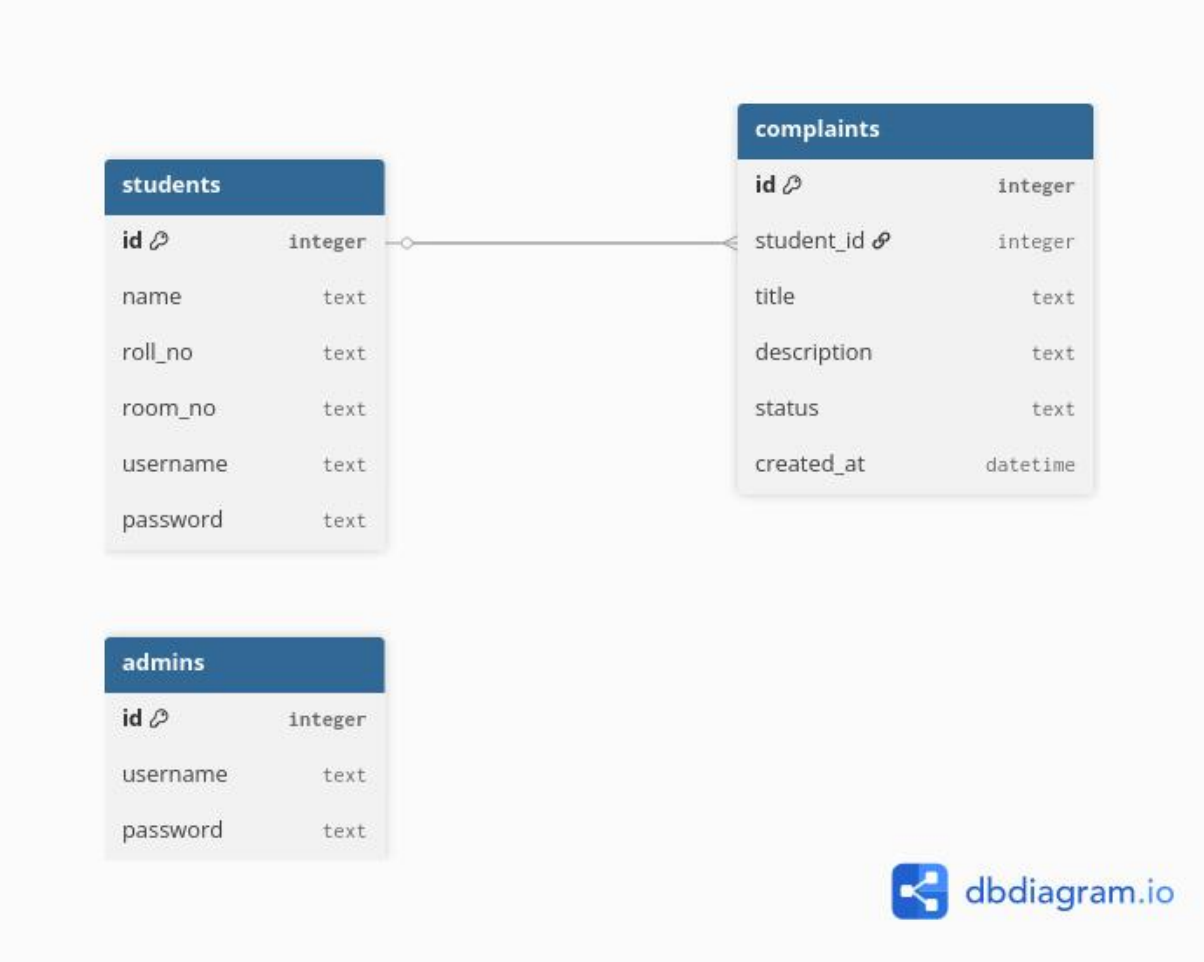
The system improves efficiency, data consistency, and security.

7. Database Design

7.1 Entities Identified

1. Admin
2. Student
3. Complaint

7.2 ER Diagram



Entities and Relationships:

- Admin manages system
- Student submits complaint
- Complaint belongs to one student

7.3 Relational Schema

ADMINS

```
CREATE TABLE IF NOT EXISTS admins (
                                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                                    username TEXT,
                                    password TEXT
);

```

STUDENTS

```
CREATE TABLE IF NOT EXISTS students (
                                      id INTEGER PRIMARY KEY AUTOINCREMENT,
                                      name TEXT,
                                      roll_no TEXT,
                                      room_no TEXT,
                                      username TEXT,
                                      password TEXT
);

```

COMPLAINTS

```
CREATE TABLE IF NOT EXISTS complaints (
                                        id INTEGER PRIMARY KEY AUTOINCREMENT,
                                        student_id INTEGER,
                                        title TEXT,
                                        description TEXT,
                                        status TEXT DEFAULT 'Pending',
                                        created_at DATETIME DEFAULT
                                        CURRENT_TIMESTAMP,
                                        FOREIGN KEY (student_id) REFERENCES
                                        students(id)
    );
```

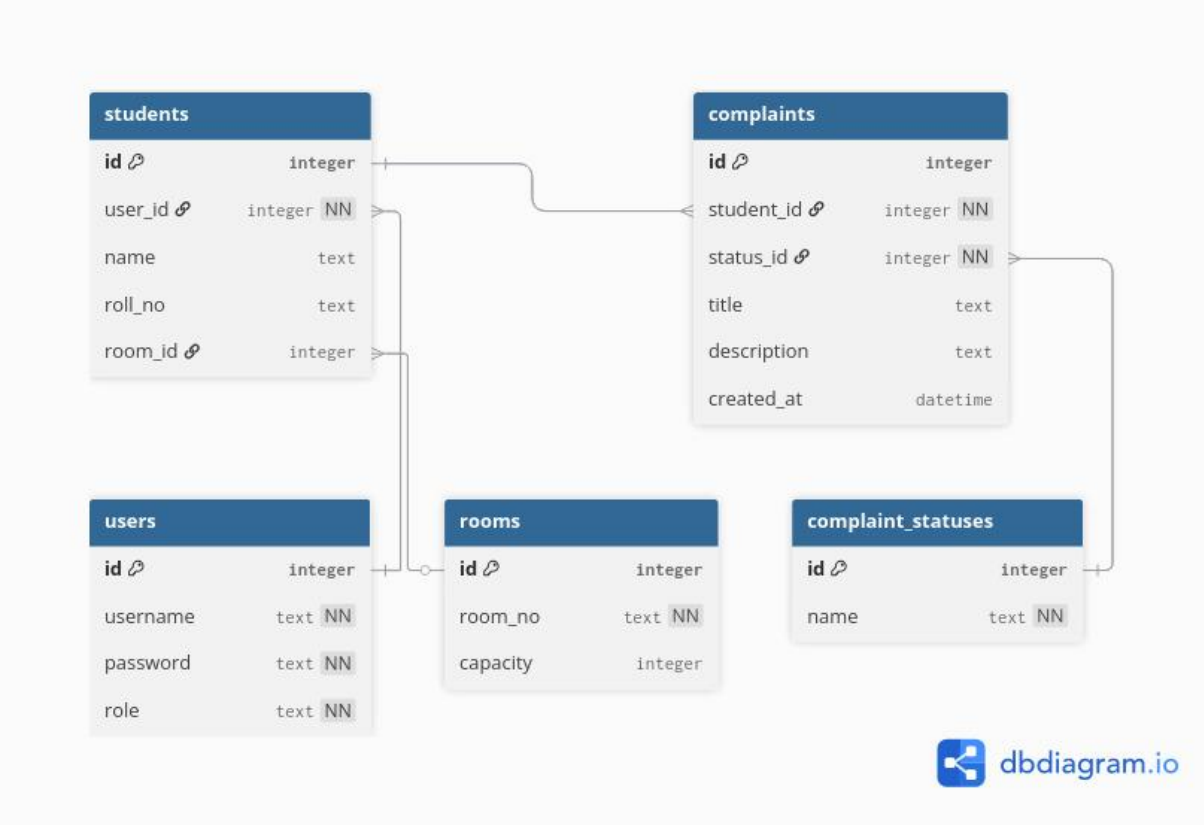
8. Normalization

Functional Dependencies

Admins:
id → username, password

Students:
id → name, roll_no, room_no, username, password

Complaints:
id → student_id, title, description, status, created_at



Third Normal Form (3NF)

- No transitive dependency.
- All non-key attributes depend only on the primary key.

Therefore, the database is normalized up to 3NF.

9. Database Implementation

9.1 SQL Implementation

DDL and DML Commands Used

```sql
CREATE TABLE IF NOT EXISTS admins (
                                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                                    username TEXT,
                                    password TEXT
);

INSERT INTO admins (username, password)
VALUES ('admin', '123');

CREATE TABLE IF NOT EXISTS students (
                                      id INTEGER PRIMARY KEY AUTOINCREMENT,
                                      name TEXT,
                                      roll_no TEXT,
                                      room_no TEXT,
                                      username TEXT,
                                      password TEXT
);

CREATE TABLE IF NOT EXISTS complaints (
                                        id INTEGER PRIMARY KEY AUTOINCREMENT,
                                        student_id INTEGER,
                                        title TEXT,
                                        description TEXT,
                                        status TEXT DEFAULT 'Pending',
                                        created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
                                        FOREIGN KEY (student_id) REFERENCES students(id)
    );
```

UPDATE Query

```sql
UPDATE complaints
SET status = 'Resolved'
WHERE id = 1;
```

9.2 Backend Logic Implementation

Since SQLite does not support PL/SQL, application logic is implemented using Go programming language.

- Authentication logic
- Complaint insertion
- Complaint status update
- Database connectivity

Database connection is handled in `db.go`:

```
db, err = sql.Open("sqlite3", "./hostel.db")
```

Error handling ensures proper database operation.

10. Tools & Technologies Used

- SQLite Database
- SQL
- Go (Golang)
- HTML
- CSS
- Bootstrap

11. Expected Outcomes

- Properly structured and normalized database
- Secure authentication system
- Efficient complaint tracking
- Improved data consistency
- Reduced manual errors