BIRMINGHAM CITY University

DIG5127 Database and Web Application Development

Milestone 2- ERM proposal

Student Name: Ankit Mahato
Student ID: 24128422
Date: 30/12/2024
Wordcount: 987
Pagecount: 11

Student ID:- 24128422

Content

Student ID:- 24128422

Introduction

This proposal outlines the design and features of a Boys' Hostel Management System, an online tool aimed at simplifying hostel operations. We want to create an easy-to-use experience for administrators, residents, and guardians, providing effective room assignments, complaint handling, billing, and tailored services to improve hostel life.

Application Overview

The Boys' Hostel Management System will function as an online platform to replace the traditional registration process on physical register books. It will offer detailed hosteller profiles, efficient complaint tracking, and secure billing options. Designed for administrators, hostellers, and guardians, the system ensures streamlined management of room allocations, dietary preferences, and other hostel activities, providing a modern and user-friendly alternative to outdated manual methods.

Users

The Boys' Hostel Management System is made for these user groups:

- **Hostellers**: Residents who can log in to update their profiles, file complaints, and check billing information.
- **Administrators**: Authorized staff who log in to manage hosteller registrations, assign rooms, handle complaints, and oversee billing.

Functionality

The main features of the Boys' Hostel Management System include:

- **User Account Management**: Easy setup for hosteller and admin accounts, along with profile updates.
- **Room Allocation Managemen**t: Smart room assignments based on what's available and user preferences, with real-time changes.
- **Complaint Managemen**t: Hostellers can file complaints publicly or privately, keeping personal issues confidential if desired. Admins are responsible for resolving and updating the status of these complaints.
- **Billing System**: Safe and effective handling of invoices, payments, and tracking their status.
- **Voting System**: Hostellers can vote on public complaints to help prioritize which issues need attention.

- **Guardian Information**: Hostellers can connect with guardians for notifications and updates.
- **Administrative Controls**: Admins can efficiently oversee hosteller registrations, room assignments, billing, and complaints.
- **Reports and Insights**: Create reports based on data for hostel occupancy, complaints, and payment statuses.

This system aims to modernize and simplify traditional manual processes.

Entity Relationship Model (ERM)

The Entity Relationship Model (ERM) serves as a graphical depiction of the interconnections among entities within a database. It offers a structured approach to comprehending the relationships and dependencies that exist between different data elements. In the context of the Boys' Hostel Management System, the ERM acts as a fundamental basis for constructing a database architecture that effectively oversees hosteller information, room assignments, grievances, financial transactions, and administrative operations.

Entities in the ERD

1. **admins**
    - **Attributes**: `adminID`, `email`, `password`.
    - **Description**: Admins manage hosteller registrations, room allocations, billing, and complaints.
2. **Hostellers**
    - **Attributes**: `hostellerID`, `name`, `email`, `password`, `roomNumber`, `dietaryPreferences`.
    - **Description**: Represents residents of the hostel. They can log in to manage their profiles, submit complaints, and view billing details.
3. **Rooms**
    - **Attributes**: `roomID`, `roomType`, `capacity`, `status`.
    - **Description**: Stores details about available rooms, including type (e.g., single, double) and occupancy status.
4. **Room Allocation**
    - **Attributes**: `allocationID`, `hostellerID`, `roomID`, `startDate`, `endDate`.
    - **Description**: Tracks which hosteller is allocated to which room and for how long.

5. **Complaints**
    - **Attributes**: `complaintID`, `hostellerID`, `description`, `status`, `visibility`.
    - **Description**: Allows hostellers to submit complaints, either publicly or privately to the admin. Private complaints remain anonymous.
6. **Votes**
    - **Attributes**: `voteID`, `complaintID`, `hostellerID`, `voteType`, `voteDate`.
    - **Description**: Tracks votes (upvotes/downvotes) submitted by hostellers on public complaints.
7. **Billing**
    - **Attributes**: `billID`, `hostellerID`, `amount`, `dueDate`, `paymentStatus`.
    - **Description**: Manages invoices and payment tracking for room fees and other charges.
8. **Guardians**
    - **Attributes**: `guardianID`, `hostellerID`, `name`, `contactInfo`.
    - **Description**: Stores information about guardians who can receive updates or alerts about their hosteller.

The ERM ensures that all data relationships, such as one-to-many connections between hostellers and complaints or rooms and room allocations, are clearly defined to streamline the system's operations.

Student ID:- 24128422

Entities	Attributes	Data Types	Constraints
admins	adminID	INT	PRIMARY KEY, AUTO_INCREMENT
	email	VARCHAR(255)	UNIQUE, NOT NULL
	password	VARCHAR(255)	NOT NULL
hostellers	hostellerID	INT	PRIMARY KEY, AUTO_INCREMENT
	name	VARCHAR(255)	NOT NULL
	email	VARCHAR(255)	UNIQUE, NOT NULL
	password	VARCHAR(255)	NOT NULL
	roomNumber	INT	FOREIGN KEY (References `Rooms.roomID`), NULLABLE
	dietaryPreferences	TEXT	NULLABLE
rooms	roomID	INT	PRIMARY KEY, AUTO_INCREMENT
	roomType	ENUM('Single', 'Double', 'Suite')	NOT NULL
	capacity	INT	NOT NULL
	status	ENUM('Available', 'Occupied', 'Under Maintenance')	NOT NULL
roomAllocation	allocationID	INT	PRIMARY KEY, AUTO_INCREMENT
	hostellerID	INT	FOREIGN KEY (References `Hostellers.hostellerI

Student ID:- 24128422

			D`), NOT NULL
	roomID	INT	FOREIGN KEY (References `Rooms.roomID`), NOT NULL
	startDate	DATE	NOT NULL
	endDate	DATE	NULLABLE
complaints	complaintID	INT	PRIMARY KEY, AUTO_INCREMENT
	hostellerID	INT	FOREIGN KEY (References `Hostellers.hostellerID`), NOT NULL
	description	TEXT	NOT NULL
	status	ENUM('Pending', 'Resolved', 'Dismissed')	DEFAULT 'Pending', NOT NULL
	visibility	ENUM('Public', 'Private')	NOT NULL
votes	voteID	INT	PRIMARY KEY, AUTO_INCREMENT
	complaintID	INT	FOREIGN KEY (References `Complaints.complaintID`), NOT NULL
	hostellerID	INT	FOREIGN KEY (References `Hostellers.hostellerID`), NOT NULL
	voteType	ENUM('Upvote', 'Downvote')	NOT NULL
	voteDate	DATETIME	DEFAULT CURRENT_TIMEST

Student ID:- 24128422

			AMP
billing	billID	INT	PRIMARY KEY, AUTO_INCREMENT
	hostellerID	INT	FOREIGN KEY (References `Hostellers.hostellerID`), NOT NULL
	amount	DECIMAL(10, 2)	NOT NULL
	dueDate	DATE	NOT NULL
	paymentStatus	ENUM('Paid', 'Unpaid', 'Overdue')	DEFAULT 'Unpaid', NOT NULL
guardians	guardianID	INT	PRIMARY KEY, AUTO_INCREMENT
	hostellerID	INT	FOREIGN KEY (References `Hostellers.hostellerID`), NOT NULL
	name	VARCHAR(255)	NOT NULL
	contactInfo	VARCHAR(255)	NOT NULL

Student ID:- 24128422

Relationships

1. **admins -> Rooms**:
    - Relationship: **One-to-Many**
    - `userID` in **admins** is a **Foreign Key (FK)** in **rooms**.
    - An admin manages multiple rooms.
2. **hostellers -> billing**:
    - Relationship: **One-to-Many**
    - `userID` in **hostellers** is a **Foreign Key (FK)** in **billing**.
    - A hosteller can have multiple bills associated.
3. **hostellers -> complaints**:
    - Relationship: **One-to-Many**
    - `userID` in **hostellers** is a **Foreign Key (FK)** in **complaints**.
    - A hosteller can file multiple complaints.
4. **complaints -> complaintVotes**:
    - Relationship: **One-to-Many**
    - `complaintID` in **complaints** is a **Foreign Key (FK)** in **ComplaintVotes**.
    - Each complaint can receive multiple votes.
5. **hostellers -> complaintVotes**:
    - Relationship: **One-to-Many**
    - `userID` in **hostellers** is a **Foreign Key (FK)** in **complaintVotes**.
    - A hosteller can vote on multiple complaints.
6. **hostellers -> guardian**:
    - Relationship: **One-to-One**
    - `userID` in **hostellers** is a **Foreign Key (FK)** in **guardian**.
    - Each hosteller has one associated guardian.
7. **hostellers -> roomAllocation**:
    - Relationship: **One-to-Many**
    - `userID` in **hostellers** is a **Foreign Key (FK)** in **roomAllocation**.
    - A hosteller can have multiple room allocations over time.
8. **rooms -> roomAllocation**:
    - Relationship: **One-to-Many**
    - `roomNumber` in **rooms** is a **Foreign Key (FK)** in **roomAllocation**.

    - A room can be allocated multiple times.

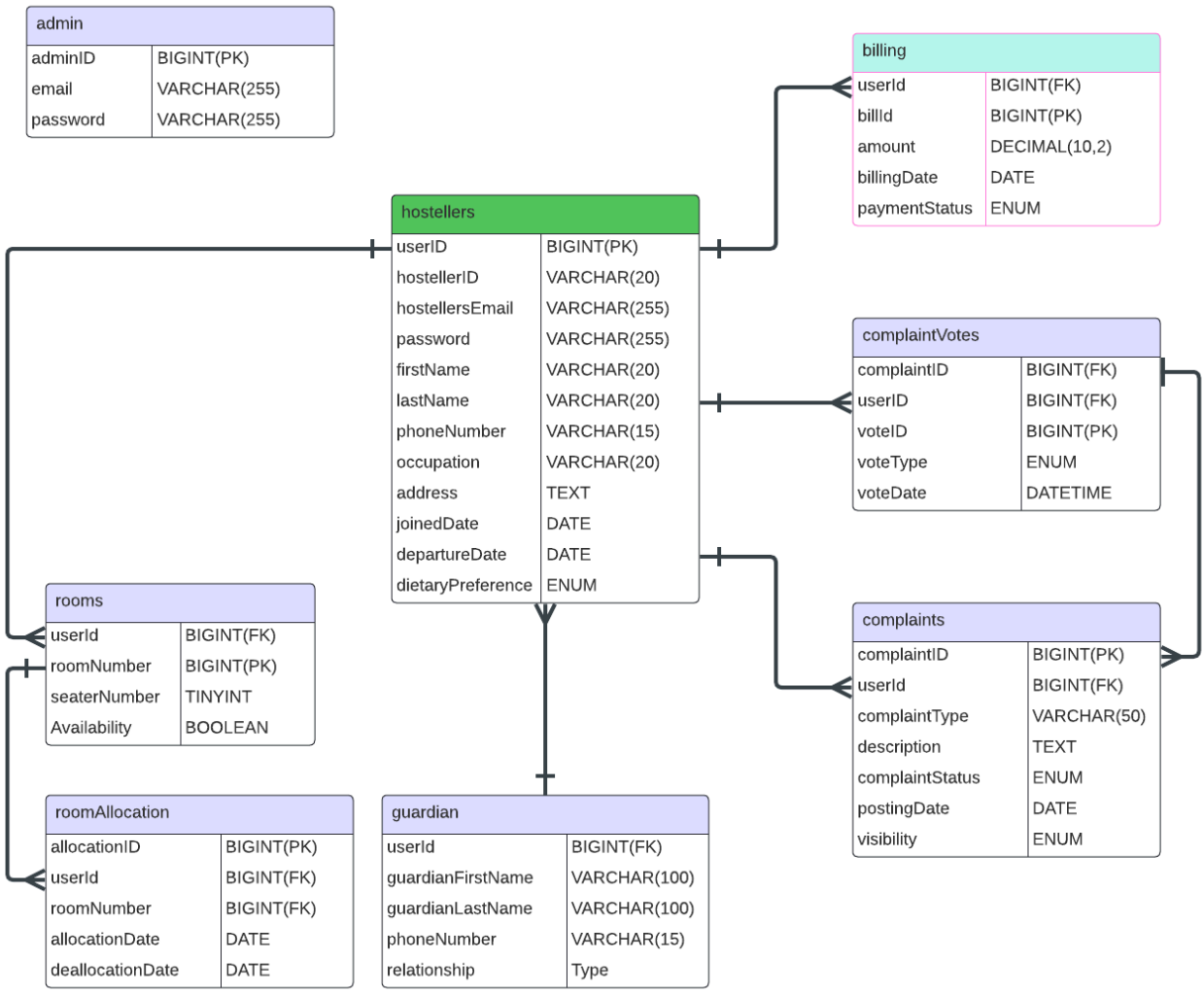
Summary of Key Foreign Keys:

- **admins -> rooms**: `userID` (FK).
- **hostellers -> billing**: `userID` (FK).
- **hostellers -> complaints**: `userID` (FK).
- **complaints -> complaintVotes**: `complaintID` (FK).
- **hostellers -> complaintVotes**: `userID` (FK).
- **hostellers -> guardian**: `userID` (FK).
- **hostellers -> roomAllocation**: `userID` (FK).
- **rooms -> roomAllocation**: `roomNumber` (FK).

If you'd like further analysis or a script for any specific task, let me know!

Student ID:- 24128422

Entity Relationship Model(ERM) Diagram



Student ID:- 24128422