ADDIS ABABA SCIENCE AND TECHNOLOGY UNIVERSITY

Department of Software Engineering

ADVANCED PROGRAMMING

Section A

Project title: **HOTEL BOOKING MANAGEMENT SYSTEM**

GROUP-9

	Name	ID
1	**Biniyam Cheru**	**ETS0296/15**
2	**Bitsuan Abate**	**ETS0328/15**
3	**Dagim Tadesse**	**ETS0343/15**
4	**Dagim Abraham**	**ETS0344/15**
5	**Michail Siameregn**	**ETS1070/14**

Submitted to: **instructor Rakeb Daba**

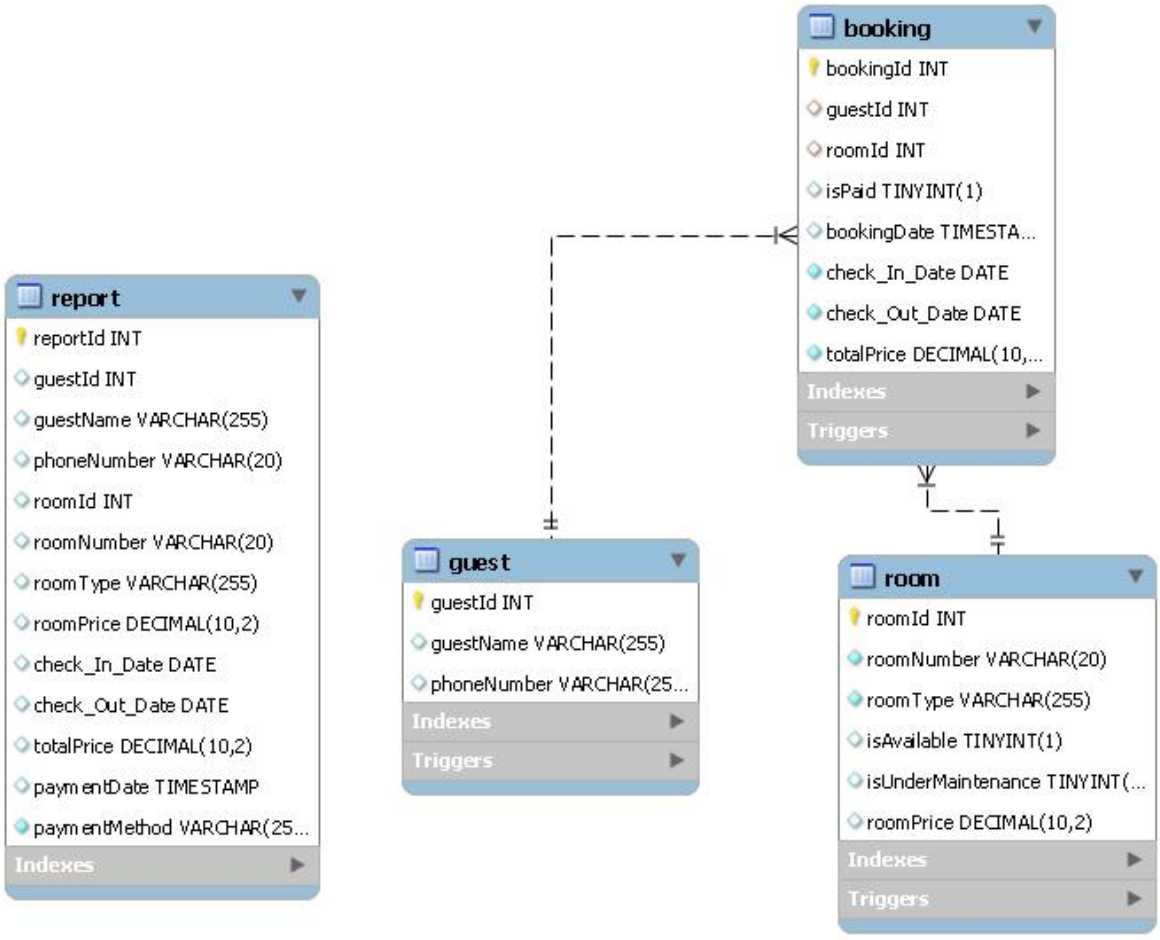
Introduction

The **Hotel Booking Management System** is a Java-based application designed to handle core operations of hotel management, including room reservations, guest management, and billing. The system supports:

- **Command-Line Interface (CLI)** for full access to all hotel operations.
- **Graphical User Interface (GUI)** prototype for demonstration and basic functionality.
- **Socket-based Networking**, allowing guest-side chat communication with the admin server.
- **Remote Method Invocation (RMI)** for remote reporting and data access.
- A **MySQL database back-end** is used to store and manage all data.

This system simulates a real-world hotel environment, offering flexibility, modularity, and extensibility.

Entity–Relationship (ER) Diagram



Entities:

- **Guest**: Stores guest information like name and contact.
- **Room**: Contains room type, number, price, availability.
- **Booking**: Maps a guest to a room for a period of time.
- **Report**: Stores completed transactions/payment logs for audit.

Relationships:

- A guest can have multiple bookings.
- A room can be booked multiple times but not simultaneously.
- Each booking has one report (generated after payment).
- Report table isn't connected to any other table because it is a payment history and should be affected by other tables.

Schema

Guest

```
create table guest(
guestId int AUTO_INCREMENT PRIMARY KEY,
guestName varchar(255),
phoneNumber varchar(255)
);
```

Room

```
CREATE TABLE room (
    roomId INT PRIMARY KEY AUTO_INCREMENT,
    roomNumber VARCHAR(20) NOT NULL UNIQUE,
    roomType varchar(255) NOT NULL,
    roomPrice DECIMAL(10,2),
    isAvailable BOOLEAN DEFAULT TRUE,
    isUnderMaintenance BOOLEAN DEFAULT FALSE
);
```

Booking

```
CREATE TABLE booking (
    bookingId INT PRIMARY KEY AUTO_INCREMENT,
    guestId INT,
    roomId INT,
    isPaid BOOLEAN DEFAULT FALSE,
    bookingDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    check_In_Date DATE NOT NULL,
    check_Out_Date DATE NOT NULL,
    totalPrice DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (guestId) REFERENCES guest(guestId) ON DELETE CASCADE,
    FOREIGN KEY (roomId) REFERENCES room(roomId) ON DELETE SET NULL
);
CONSTRAINT check_dates CHECK (check_Out_Date > check_In_Date);
```

Report

```sql
CREATE TABLE report(
    reportId INT PRIMARY KEY AUTO_INCREMENT,
    guestId INT,
    guestName VARCHAR(255),
    phoneNumber VARCHAR(20),
    roomId INT,
    roomNumber VARCHAR(20),
    roomType VARCHAR(255),
    roomPrice DECIMAL(10,2),
    check_In_Date DATE,
    check_Out_Date DATE,
    totalPrice DECIMAL(10,2),
    paymentDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    paymentMethod VARCHAR(255) NOT NULL
);
```

Relation with Normalization

Table	1NF	2NF	3NF
Guest	✅	✅	✅
Booking	✅	✅	✅
Report	✅	✅	✅

Normalization Justification:

- All tables contain **atomic values** (1NF).
- All non-key columns are fully dependent on the **primary key** (2NF).
- No **transitive dependencies** exist in any table (3NF).
- Repeating data (e.g., roomType, guestName) is present in **report** by design for historical records.

Key Functions

❖ **CLI Core Features (HotelManagement.java)**

- Add, update, delete, and list:

  I. Rooms
  II. Guests
  III. Bookings

- Billing & payment
- Reports (Daily Revenue, Guest Summary)
- Better check-in-date and check-out-date handling
- Validation of date

- **GUI Features (MainGUI.java, Panels)**
  - Login screen with hardcoded credentials
  - Panels for:
    - ✓ Managing guests, rooms, bookings.
  - Report and billing handling
  - Guest Chat access via button
- **Networking**
  - AdminServer.java: Runs socket server for chat
  - GuestHandler.java: Handles each guest in its thread
  - MessageProtocol.java: Processes guest input (basic NLP simulation)
  - GuestClient.java: CLI client
  - ChatPanel.java: GUI-based chat window
- **RMI**
  - Report interface: Defines remote methods
  - ReportImp.java: Implements report logic
  - RMIServer.java: Registers ReportImp
  - Remote functions:
    - ✓ Read all reports
    - ✓ Insert new report
    - ✓ Get total daily revenue
    - ✓ Count payments for a date

Conclusion

The Hotel Booking Management System demonstrates a full-stack software solution in Java, blending:

- Traditional database interaction (JDBC)
- Modern distributed computing techniques (Sockets, RMI)
- User-friendly interfaces (CLI and Swing GUI)

It is modular and extensible, making it ideal for educational purposes and a great foundation for a real-world product. The layered structure—GUI, network, business logic, and data access—ensures **separation of concerns** and maintainability.