**Group 4.**

Restaurant Booking website

Name:

**1. Design the Database (Using JDBC)**

* Define the database structure. Some recommended tables:
  + users (id, name, email, password, etc.).
  + restaurants (id, name, address, cuisine type, etc.).
  + bookings (id, user\_id, restaurant\_id, booking\_date, number\_of\_people, etc.).
* Create the database using **MySQL** or another relational database management system.
* Connect your application to the database using **JDBC**.

**2. Create Servlets for Backend Logic (Using Servlets)**

* Create servlets to handle user requests:
  + **UserServlet** to manage registration, login, and user profiles.
  + **RestaurantServlet** to list and display restaurant details.
  + **BookingServlet** to manage booking creation, modification, and cancellation.
* Use **JDBC** in the servlets to connect to and manipulate the database.

**3. Develop JSP Pages for the User Interface (Using JSP)**

* Create **JSP** pages for the front-end interface:
  + index.jsp: homepage where users can search for restaurants.
  + login.jsp and register.jsp: forms for login and registration.
  + restaurantDetails.jsp: displays restaurant details and booking form.
  + myBookings.jsp: allows users to view their reservations.
* Use JSTL or EL (Expression Language) to display dynamic data from the servlets on the JSP pages.

*5. (Set Up Navigation Control)*

* In your application's web.xml file, configure the servlet mappings to route HTTP requests (GET and POST) to the appropriate servlets.

**6. Implement Authentication and Authorization**

* Implement authentication for users who want to make reservations, using sessions.
* Protect certain pages (like booking) so that only authenticated users can access them.

**7. Test and Debug**

* Ensure all features are properly implemented:
  + User registration and login.
  + Restaurant viewing.
  + Creating, modifying, and canceling reservations.
* Use logging (with **Log4j** or **SLF4J**) to track and fix errors.

**8. Deploy the Application**

* Once development and testing are complete, deploy the application to a production server (e.g., Apache Tomcat).

**Technologies Used:**

* **JSP**: for building the user interface.
* **Servlets**: for handling user requests and business logic.
* **JDBC**: for database connectivity and manipulation.

**Workflow Overview:**

1. The user interacts with the JSP pages.
2. Requests are sent to the servlets.
3. Servlets interact with the database through JDBC.
4. The response is sent back to the client via JSP.

**SCHEMA DESIGN**

***-- Create the database***

CREATE DATABASE IF NOT EXISTS single\_restaurant\_booking;

**Users Table**

For customer and admin details:

* **id**: Primary Key (int, auto-increment)
* **name**: User's full name (VARCHAR)
* **email**: Unique identifier (VARCHAR)
* **password**: Hashed password (VARCHAR)
* **phone\_number**: Contact number (VARCHAR)
* **role**: Defines if the user is an admin or a regular user (VARCHAR, e.g., "admin" or "customer")
* **created\_at**: Timestamp of when the account was created (TIMESTAMP)

***-- Create Users Table***

CREATE TABLE IF NOT EXISTS users (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL,

phone\_number VARCHAR(20),

role VARCHAR(50) CHECK (role IN ('admin', 'customer')) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP);

**Restaurant Table**

Details of the single restaurant:

* **id**: Primary Key (int, auto-increment, though it will only have one entry)
* **name**: Restaurant name (VARCHAR)
* **address**: Full address (VARCHAR)
* **cuisine\_type**: Type of cuisine (VARCHAR, e.g., "Italian", "Chinese", etc.)
* **phone\_number**: Restaurant's contact number (VARCHAR)
* **opening\_time**: Opening time (TIME)
* **closing\_time**: Closing time (TIME)
* **created\_at**: Timestamp of when the restaurant was added (TIMESTAM
* **max\_daily\_capacity**: The maximum number of people allowed to book each day (int).

***-- Create Restaurant Table***

CREATE TABLE IF NOT EXISTS restaurant (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

address VARCHAR(255) NOT NULL,

cuisine\_type VARCHAR(100) NOT NULL,

phone\_number VARCHAR(20),

opening\_time TIME NOT NULL,

closing\_time TIME NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

max\_daily\_capacity INT NOT NULL CHECK (max\_daily\_capacity > 0));

**Tables Table**

This new table will manage the restaurant's tables and their capacity.

* + **id**: Primary Key (int, auto-increment)
  + **table\_number**: Unique identifier for each table (VARCHAR)
  + **capacity**: Maximum number of people the table can accommodate (int)
  + **status**: Indicates if the table is available or booked (VARCHAR, e.g., "available", "booked")
  + **created\_at**: Timestamp of when the table was added (TIMESTAMP)

***-- Create Tables Table***

CREATE TABLE IF NOT EXISTS tables (

id INT AUTO\_INCREMENT PRIMARY KEY,

table\_number VARCHAR(50) UNIQUE NOT NULL,

capacity INT NOT NULL CHECK (capacity > 0),

status VARCHAR(50) DEFAULT 'available' CHECK (status IN ('available', 'booked')),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP);

**Daily\_Capacity Table**

This table will store the daily capacity settings managed by the admin.

* + **id**: Primary Key (int, auto-increment)
  + **date**: The date for which the capacity is set (DATE)
  + **total\_capacity**: The maximum number of people allowed for that day (int)
  + **created\_at**: Timestamp of when the record was created (TIMESTAMP)

***-- Create Daily\_Capacity Table***

CREATE TABLE IF NOT EXISTS daily\_capacity (

id INT AUTO\_INCREMENT PRIMARY KEY,

date DATE UNIQUE NOT NULL,

total\_capacity INT NOT NULL CHECK (total\_capacity > 0),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP);

**Bookings Table**:

Updated to include an additional check to ensure the total bookings do not exceed the total\_capacity for the day.

* + **id**: Primary Key (int, auto-increment)
  + **user\_id**: Foreign Key referencing users(id)
  + **table\_id**: Foreign Key referencing tables(id)
  + **booking\_date**: Date and time of the booking (DATETIME)
  + **number\_of\_people**: Number of people for the reservation (int)
  + **status**: Status of the booking (VARCHAR, e.g., "pending", "confirmed", "canceled")
  + **created\_at**: Timestamp of booking creation (TIMESTAMP)

***-- Create Bookings Table***

CREATE TABLE IF NOT EXISTS bookings (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

table\_id INT NOT NULL,

booking\_date DATETIME NOT NULL,

number\_of\_people INT NOT NULL CHECK (number\_of\_people > 0),

status VARCHAR(50) CHECK (status IN ('pending', 'confirmed', 'canceled')) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE,

FOREIGN KEY (table\_id) REFERENCES tables(id) ON DELETE CASCADE);

**Reviews Table**

optional for user feedback:

* **id**: Primary Key (int, auto-increment)
* **user\_id**: Foreign Key referencing users(id)
* **rating**: Rating out of 5 (int)
* **comment**: User review/comment (TEXT)
* **created\_at**: Timestamp of when the review was posted (TIMESTAMP)

***-- Create Reviews Table***

CREATE TABLE IF NOT EXISTS reviews (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

rating INT CHECK (rating >= 1 AND rating <= 5),

comment TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE);

A screenshot of a computer

Description automatically generated

**User:**  root

**Password:** Root

PassA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Using the SQL script we created the DB.

**Download the MySQL JDBC Driver**

**1. Download MySQL Connector/J:**

* Go to the [MySQL official site](https://dev.mysql.com/downloads/connector/j/) and download MySQL Connector/J.
* Once downloaded, extract the contents and locate the mysql-connector-java-x.x.xx.jar file (the version will depend on the latest available at the time).

**Add MySQL JDBC Driver to Tomcat and Your Project**

Since you are using **Tomcat**, you can add the JDBC driver in two ways:

**Add MySQL Connector JAR to Your Project's WEB-INF/lib Folder**

1. **Add MySQL Connector JAR to Your Project**:
   * Copy the mysql-connector-java-x.x.xx.jar file into your project’s WEB-INF/lib folder. For example:

vbnet

Copy code

YourProject/

├── WEB-INF/

│ ├── lib/

│ │ └── mysql-connector-java-x.x.xx.jar