Course CSI2132 Databases I

Course Project - Deliverable 2

Group #15
Samuel Krutis #300229769
Alec Bazinet # 300261019

a. The DBMS and the programming languages that you have used in your implementation of the application.

Database Management System (DBMS):

- **DBMS Used:** MySQL
- **Version:** 8.0
- MySQL was chosen due to its robustness, ease of integration with Java-based backend systems, and wide community support.

Backend Development:

- **Programming Language:** Java
- **Java Version:** JDK 21
- **Framework:** Spring Boot
- **Build Tool:** Gradle
- We used Spring Boot for building a modular, RESTful API server that communicates with the MySQL database. Gradle was used for managing dependencies and building the project.

Frontend Development:

- **Programming Language:** JavaScript
- **Library:** React

- No additional frontend CSS framework was used; the UI was built using standard HTML, CSS, and React components.

Client-Server Communication:

- We implemented **RESTful APIs** for communication between the frontend and backend, allowing for a clean separation of concerns and efficient data exchange in JSON format.

b. Specific steps to guide someone to install your applications

Frontend Installation and Execution

1. Open a terminal window.
2. Navigate to the frontend directory. From the root of the project, run:
   ```
   cd web-app
   ```
3. Install the frontend dependencies by running:
   ```
   npm install
   ```
4. Start the frontend development server by executing:
   ```
   npm start
   ```

This will launch the frontend application locally, typically accessible at `http://localhost:3000/`.

Backend Installation and Execution

1. Open a new terminal window.
2. Navigate to the backend directory. From the root of the project, run:
   ```
   cd backend/demo
   ```
3. Build the backend using Gradle:
   - On **Linux/macOS**, run:
     ```
     ./gradlew clean build
     ```

- On **Windows**, run:
  `.\gradlew clean build`

4. Once the build completes, run the backend server:

   - On **Linux/macOS**, run:
     `./gradlew bootRun`

   - On **Windows**, run:
     `.\gradlew bootRun`

The backend server will start locally and is typically accessible at `http://localhost:8080/`

c. A list with the DDLs that create your database

The full database creation file can be found in the SQL directory of the project code
CSI2132-Project/SQL/DB_Builder.session.sql

```
-- @block
CREATE TABLE hotel_chain(
   chain_id INT PRIMARY KEY AUTO_INCREMENT,
   chain_name VARCHAR(255),
   chain_address VARCHAR(255),
   number_of_hotels INT,
   email_addresses TEXT,
   phone_numbers TEXT
)

CREATE TABLE hotel(
   hotel_id INT PRIMARY KEY AUTO_INCREMENT,
   chain_id INT NOT NULL,
   FOREIGN KEY (chain_id) REFERENCES hotel_chain(chain_id),
   hotel_name VARCHAR(255),
   rating INT,
   hotel_address VARCHAR(255),
   city VARCHAR(255),
   state VARCHAR(255),
   amount_of_rooms INT,
   contact_email VARCHAR(255) REFERENCES hotel_chain(email_addresses),
   contact_phone VARCHAR(255) REFERENCES hotel_chain(phone_numbers),
   manager_id INT
)

CREATE TABLE employee(
   employee_id INT PRIMARY KEY AUTO_INCREMENT,
   hotel_id INT NOT NULL,
   FOREIGN KEY (hotel_id) REFERENCES hotel(hotel_id),
   employee_name VARCHAR(255),
   employee_address VARCHAR(255),
   SIN_num INT,
   employee_position VARCHAR(255)
)

CREATE TABLE room(
   room_id INT PRIMARY KEY AUTO_INCREMENT,
   hotel_id INT NOT NULL,
   FOREIGN KEY (hotel_id) REFERENCES hotel(hotel_id),
   price DECIMAL(10,2),
   view VARCHAR(255),
   amentities TEXT,
   extendable BOOLEAN,
   capacity INT,
   damages TEXT
)

CREATE TABLE customer(
   customer_id INT PRIMARY KEY AUTO_INCREMENT,
   id_type VARCHAR(255),
   customer_address VARCHAR(255),
   customer_name VARCHAR(255),
   registration_date DATE,
   id_number VARCHAR(255)
)

CREATE TABLE booking(
   booking_id INT PRIMARY KEY AUTO_INCREMENT,
   room_id INT NOT NULL,
   customer_id INT NOT NULL,
   FOREIGN KEY (customer_id) REFERENCES customer(customer_id),
   FOREIGN KEY (room_id) REFERENCES room(room_id),
   booking_date DATE,
   checkin_date DATE,
   checkout_date DATE,
   status VARCHAR(255)
)
```

```
CREATE TABLE renting(
    renting_id INT PRIMARY KEY AUTO_INCREMENT,
    room_id INT NOT NULL,
    customer_id INT NOT NULL,
    FOREIGN KEY (customer_id) REFERENCES customer(customer_id),
    FOREIGN KEY (room_id) REFERENCES room(room_id),
    start_date DATE,
    end_date DATE,
    payment_status VARCHAR(255)
)

CREATE TABLE archive(
    archive_id INT PRIMARY KEY AUTO_INCREMENT,
    booking_id INT,
    renting_id INT,
    FOREIGN KEY (booking_id) REFERENCES booking(booking_id),
    FOREIGN KEY (renting_id) REFERENCES renting(renting_id),
    checkin_date DATE REFERENCES booking(checkin_date),
    checkout_date DATE REFERENCES booking(checkout_date),
    booking_date DATE REFERENCES booking(booking_date),
    start_date DATE REFERENCES renting(start_date),
    end_date DATE REFERENCES renting(end_date)
)
```