Gisma University of Applied Sciences

Assessment Submission Form

Student Number (If this is group work, please include the student numbers of all group participants)	GH1024695 Mahdieh Rajabi
Assessment Title	Fullstack web application
Module Code	M607
Module Title	Computer science application lab
Module Tutor	Alireza Mahmoud
Date Submitted	19.12.2024

Declaration of Authorship

I declare that all material in this assessment is my own work except where there is clear acknowledgement and appropriate reference to the work of others.

I fully understand that the unacknowledged inclusion of another person's writings or ideas or works in this work may be considered plagiarism and that, should a formal investigation process confirms the allegation, I would be subject to the penalties associated with plagiarism, as per GISMA Business School, University of Applied Sciences' regulations for academic misconduct.

Signed……Mahdieh Rajabi…….. Date ……19.12.2024……

Hotel booking site

(full-stack project)

GitHub link:

https://github.com/Mahdiehrajabi/Computer-science-lab.git

Abstract

The goal of this project is to provide a hotel and room management system that makes it easier to manage reservations, hotel data, and room assignments. In addition to providing safe role-based access control for administrators and ordinary users, the project tackles the difficulties in efficiently handling hotel and room data. The front end of the system is built with React.js, the backend with Node.js and Express.js, HTML, CSS and the relational database used for data management and storage is PostgreSQL. To guarantee safe and controlled access, JWT (JSON Web Tokens) are used to implement user authentication and authorization. User identification, role-based access control (enabling administrators to oversee operations), dynamic hotel and room administration (adding, modifying, and removing records), and an intuitive, responsive interface for smooth interaction are some of the project's key features. Scalability and effective data retrieval via structured database queries are features built into the system. The project's result is a useful, safe, and effective management system that streamlines administrative procedures, improves data accuracy, and offers a strong basis for future additions like reporting and booking management capabilities. The system delivers dependable performance and data consistency by utilizing PostgreSQL for relational data management, which makes it an appropriate choice for small to medium-sized hotel management companies.

Table of contents

1. Introduction……..……4
2. Objective…….…….5
3. Configuring the Development Environment and Tools…….……..6

3.1. Required tools (Backend)……..……6
3.2. Install tools……..……6
3.3. Project setup……..…….7

4. Database design……...…….8
5. Feature development……9
6. Test APIs with Postman……..….13
7. Frontend…….……..14
8. Conclusion……...15
9. References……16

1. Introduction

Effective hotel and room data management is essential in today's hospitality sector to guarantee seamless operations and outstanding guest experience. Conventional hotel information management, room assignments, and reservation systems are frequently manual and prone to mistakes, which results in inefficiencies, a lack of scalability, and challenges with data tracking and updating. It has become essential to develop a solid solution that streamlines these procedures while offering safe, role-based access control for various user types, including administrators and clients, as the need for more dynamic and automated systems grows. The increasing demand for digital transformation in the hospitality industry provides the driving force behind this initiative. Hotel managers need a system that guarantees data integrity and safe access in addition to enabling them to add, amend, and manage their hotel and room data with ease. At the same time, customers who engage with the system anticipate an intuitive interface that makes it simple for them to investigate the accommodation that is offered. To increase productivity and save on administrative costs in hotel management, these demands must be met with a scalable and safe system. In the actual world, this project is significant because it offers a fundamental system that small and medium-sized hotels and other hospitality enterprises can modify. The project makes use of relational databases like PostgreSQL and contemporary web technologies like React, Node.js, and Express.js to guarantee scalability, security, and responsiveness. Furthermore, only authorized administrators can alter sensitive data thanks to role-based access control, shielding vital company information from unwanted access. This project's main goal is to develop a hotel and room management system that enables administrators to control hotel and room information in real time. This includes establishing room characteristics like capacity and cost, adding new hotels, and updating or removing current data. To tour hotels and rooms without interfering with administrative processes, regular users are given restricted access. To enable future expansion to incorporate features like booking systems, real-time room availability, and sophisticated analytics, the system prioritizes scalability, security, and usability. In the end, this project tackles operational issues in the hotel sector and offers a framework that may be expanded for further improvements. The objective of this project is to decrease mistakes, enhance data accuracy, and enable businesses to provide better services to their clients in a more competitive market by optimizing workflows and automating data management operations.

2. Objective

This project's main goal is to create a complete, flexible hotel and room management system that simplifies administrative duties and offers a user-friendly interface. This project used React.js to create a responsive front-end interface that works well and is accessible on a range of devices, including smartphones, tablets, and PCs. To effectively handle API requests and manage business logic, the system also has a strong backend server built using Node.js and Express.js. A PostgreSQL relational database is incorporated for data management to store and arrange structured data about rooms, hotels, and people in a consistent and dependable manner. Using JSON Web Tokens (JWT) to enable role-based access control and secure user authentication is a

key objective of the project. This guarantees that important tasks, including adding, editing, or removing hotel and room data, can only be carried out by authorized administrators. Administrators can effectively manage hotel details, room attributes, and pricing thanks to the system's capability for dynamic CRUD operations. Conversely, regular users are given limited access to view the hotels and rooms that are available without affecting administrative features. Additionally, the project's design prioritizes maintainability and scalability. Future improvements including complicated booking systems, real-time tracking of room availability, and reporting features are supported by the design. By achieving these goals, the system helps small and medium-sized hospitality organizations improve their workflows, increase operational efficiency, and improve the accuracy of data management.

3. Configuring the Development Environment and Tools

3.1. Required tools (Backend)

A variety of contemporary tools and technologies are used in this project to guarantee effective development, scalability, and maintainability. The main IDE/code editor for writing and maintaining code is Visual Studio Code, which offers features like integrated terminal, debugging, and syntax highlighting for a smooth development process.

- IDE/Code editor: Visual Studio Code
- PostgreSQL database (relational)
- Language and frameworks
- Backend: Node.js with Express.js
- Frontend: React.js, HTML, CSS
- Project management tools
- GitHub for code storage and version control
- Postman for API testing

3.2. Install tools

1. Install Node.js and npm

Download from the official Node.js website.

2. Setup PostgreSQL

For PostgreSQL, installed the PostgreSQL Installer.

The project creates scalable, effective, and organized architecture that is appropriate for practical use cases by using these tools and technologies.

3.3. Project Setup

Backend

1. Create the project folder (with this command)

sets a backend project directory: mkdir hotel-booking-backend && cd hotel-booking-backend

uses a default package to start the project.JSON file: npm init -y

2. Install the required libraries:

(PostgreSQL) bcrypt jsonwebtoken dotenv cors

3. Folder structure

The structure would be like this:

3. Folder structure:

hotel-booking-backend/

├── controllers/ # API logic

├── models/ # Database models

├── routes/ # Routes

├── config/ # Settings (like database connection)

├── app.js # Main application log

├── package.json # Project information

Frontend Setup

React.js is used to design the project's frontend, which creates a responsive and dynamic user experience. For full-screen views that correspond to routes, we need components for reusable user interface elements and pages, as well as services for clean API integration and HTTP logic.

1. Create React project:

npx create-react-app hotel-booking-frontend

2. Install libraries:

npm install axios react-router-dom bootstrap

3. Folder structure:

hotel-booking-frontend/

├── src/

├── components/ # UI components

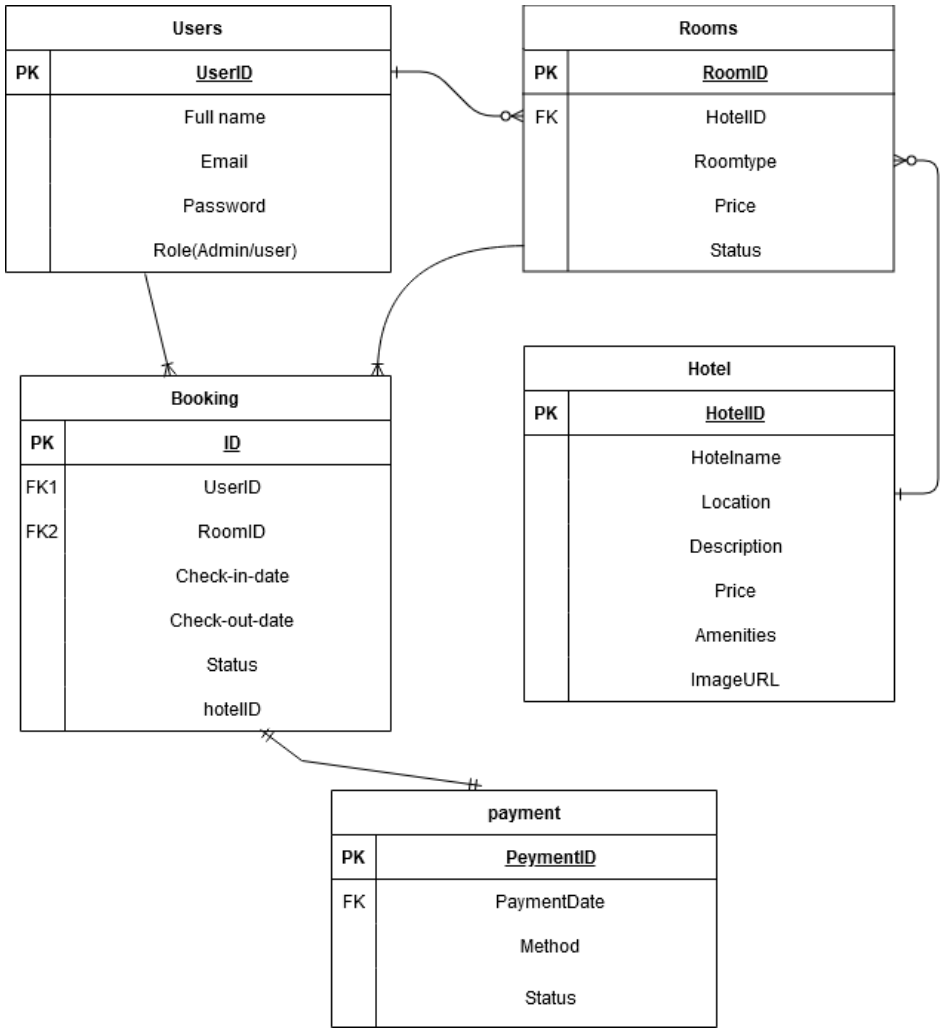
├── pages/ # Pages (like Home, Admin, Booking)

├── services/ # API files

4. Database Design

Data Structure

PostgreSQL, a relational database that facilitates organized data and table relationships, was used in the database's design. A thorough description of each table's relationships, data structure, and function can be found below.



5. Feature Development

Authentication System

The application's secure access is guaranteed by the authentication system, which enables users to register and log in. JSON Web Tokens, or JWTs, are used for safe token-based authentication.

Backend: Main paths: /signup: Registration.

Users enter their role, password, email address, and name.
Before being stored in the database, passwords are safely hashed using bcrypt.

/login: User login.

creates a JWT token for allowed users and verifies user credentials (password and email).

Admin Panel

Administrators can control hotel and room data using the Admin Panel. This entails using backend APIs to carry out CRUD actions on hotels and rooms and managing and displaying this data in the frontend.

API paths:

/hotels: Hotels CRUD.

/rooms: Room management.

Frontend: The frontend requires a React component to handle hotel data in order to communicate with the backend APIs. The admin can add, update, and remove hotels using this component, which also retrieves hotel data and presents it in a table.

Booking System

Users can make, view, and modify reservations using the booking system. It guarantees efficient administration of booking-related data and links users, rooms, and hotels in the database.
All CRUD actions for booking management are handled by the backend API path /bookings.

Backend: Path:

/bookings: Example:

Creating a Booking (POST /bookings) enables users to reserve a room by indicating the dates they would like to check in and out.

View Bookings (GET /bookings) retrieves all reservations for administrators or the person that is currently logged in.

Modify a Reservation (PUT /bookings/:id)
allows users or administrators to change the dates and status of reservations.

Remove of a reservation (DELETE /bookings/:id)
enables booking cancellation.

Google Maps API Integration

Integrating Google Maps API, by registering the API key in Google Cloud Console, may improve the user experience by showing hotel locations on a map.

Setting up a Node.js project

Create a project and initial settings:

1. Create a project folder and enter it with this command:

mkdir hotel-booking-backend

cd hotel-booking-backend

Then create a package.json file:

npm init -y

This command creates a package.json file, which contains the initial information of the project.

Installation of necessary dependencies

With the following command installation is possible:

npm install express mongoose dotenv cors jsonwebtoken bcrypt

The important dependencies are express, dotenv, cors, jsonwebtoken, bcrypt.

Express is a Node.js web framework that is lightweight and adaptable, making it easier to create backend servers and APIs. It facilitates effective server-side logic, middleware, and route management. In more precise, it handles HTTP requests and responses by defining API pathways like /signup, /login, /hotels, and /bookings.

Next is **Dotenv**, a tool for safely managing environment variables. Sensitive data, such API keys, JWT secrets, and database credentials, are not hardcoded into your code but are instead stored in.env files. In addition to making the project more secure and adaptable, it safeguards sensitive data.

When the front end and backend are hosted on separate domains or ports (for example, the front end on localhost:3000 and the backend on localhost:3001), a middleware called **CORS (Cross-Origin Resource Sharing)** enables communication between them. Due to security regulations, the browser will reject frontend queries to your backend if CORS is not used.

The management of user permission and authentication is done by the **JSON Web Token (JWT)**. To confirm the user's identity, it creates secure tokens at login and sends them along with

further requests. To manage user sessions without keeping user data on the server, it offers a safe, stateless solution.

Bcrypt is used to safely hash passwords before putting them in a database.
Passwords can be made unreadable using hashing, even if the database is compromised.

In the project folder, following commands should run:

npm install pg pg-hstore sequelize

PostgreSQL with Sequelize **pg (PostgreSQL Client for Node.js)** Dependencies
The main PostgreSQL client for Node.js is the pg library. It makes it easier for you to connect your PostgreSQL database to your Node.js backend server so you can run queries and communicate with the database.
It serves as PostgreSQL's foundational library, facilitating direct communication and SQL execution.

Pg-Hstore is a PostgreSQL efficiency library for handling JSON data. Storing and decoding JSON objects for storage is made easier with this module, as PostgreSQL supports both JSON and JSONB column types.
The pg-hstore facilitates the handling of JSON data if your application has to store JSON objects in PostgreSQL (such as user preferences or room amenities).

sequelize (the library for object-relational mapping)
For working with relational databases (such as PostgreSQL, MySQL, and SQLite), Sequelize is an ORM (Object-Relational Mapping) framework for Node.js that offers an abstraction layer. Sequelize lets you work with the database using JavaScript objects and models rather than making raw SQL queries. It makes interacting with SQL databases less complicated and supports the use of JavaScript objects for tables in model-based development. Also, it provides CRUD relationships and operations (many-to-many and one-to-many) simpler.

2. Setting up a connection to PostgreSQL

Create a database configuration file

Create a new file called config/database.js with the desired command:

mkdir config

touch config/database.js

Then add the connection code and carry out Database connection settings. In more detail, the creation of a specific configuration file (database.js) allows for the centralization of all database settings (host, login, password, etc.). It improves the code's cleanliness and accessibility.

Only the.env file, not the entire codebase, needs to be updated whenever database credentials change.

After that, Checking the database connection is essential. to detect faults related to improper credentials, settings, or network problems before executing the program. Moreover, to confirm that the program is able to connect to the appropriate database environment (development, testing, or production)

3.Using Sequelize for Data Modeling

Creating a Folder for Models with this command: mkdir models

4. Database synchronization

In the app.js file, add specific settings like:

const sequelize = require('./config/database');

const User = require('./models/User');

API routes for data management

Add user registration route: routes/auth.js

And each new route should be added to app.js

Use node app.js to launch the server.
To ensure the user is stored correctly, use Postman to test the /auth/signup path.

Now that you have a database connection, you can create more models to store data.

1. In the folder, create a file called Hotel.js, with this model, it is possible to manage and save hotel data.
2. Create a file called Room.js in the model's folder. The Room model manages rooms for each hotel, and rooms can be connected to hotels.

Syncing Database

In the app.js file, call sequelize.sync () to sync models with the database

There is force in the code that should be true and will remove and rebuild if the tables are already available. You can change or delete this option in the production environment.

*All the routes should be at app.js file.

6. Test APIs with Postman

Testing is an essential step to make sure the endpoints work as expected after the backend APIs are created and linked to the database. Developers can simulate client-server connection, make HTTP requests like GET, POST, PUT, and DELETE to the server, and check answers with Postman, a popular tool for testing APIs. Developers can test a variety of scenarios with Postman, including receiving incorrect inputs, providing valid inputs, and verifying edge situations to make sure the server handles requests appropriately and responds with the right answers. By showing comprehensive server replies, including status codes, response times, and data returned in JSON or other forms, Postman further simplifies the debugging process. Postman enables developers to easily add authorization headers to APIs that need secure access, like protected routes with JWT (JSON Web Tokens). This is especially crucial when testing role-based APIs like /hotels for CRUD operations or /bookings for reservation management, or endpoints like /signup and /login. Developers can quickly check whether users with roles have the appropriate amount of access and whether token-based authentication is operating as intended. Additionally, Postman offers a straightforward method for testing both successful operations and error-handling features, including missing arguments, erroneous payloads, and unauthorized access. Apart from manual testing, Postman facilitates the formation of collections—reusable sets of API queries. These collections can be distributed among groups and even programmed to run regular assessments, guaranteeing the system's dependability over time. Finally, before integrating with the frontend, Postman ensures that the backend APIs satisfy functional and security requirements, expedites debugging, and simplifies API testing.

Verifying that the data is appropriately stored in your PostgreSQL database comes next, after Postman testing of APIs to ensure that data is being transferred to the backend. With pgAdmin, a graphical administrative tool for PostgreSQL, you may examine the database, verify the data entered through the APIs, and check tables. This step makes sure that the queries are running correctly and that the connection between your backend server and the database is operating as it should.

Then, data should be saved there.

For Display Reservations (Get Request)

After the data has been correctly entered, you can add the Get function to show all of the reservations or filter them according to various criteria (such Hotelid or Userid).

Create the Get Get Rail to Get Reservations, obviously, in the Routes/Booking.js file, can add a new path to receiving all the reservations:

Then test in Postman is vital: URL: http: // localhost: 3000/bookings

This request will bring all the reservations in the database.

Implementation of Authentication

Implementation of an Authentication System
It is suggested that use JWT (JSON Web Token) for user authentication so that users can only access their reservations and system security is maintained. Users can access the token through this mechanism and use it to make further requests.
Steps for Identity Idgeneys (JWT) is to set up the required libraries like Installing jsonwebTok and bcryptjs (for password encryption) and make a new file named Auth.js in the Routes folder.

Then, for integrate Confirm Token (JWT) Middleware, need to build a modleware to validate the JWT token so that only users may access their reservations and data.

7. Front-end

First installed the react and with following command creates a new React project called Hotel-Booking-Frontend.

The structure of the react project folders would be

Hotel-Booking-Frontend/

├── SRC/

│ ├── Components/ # UI Components

│ ├── pages/ # different pages

│ ├── Services/ # for API requests

│ └── app.js # main file

└── Public/

Additionally, Install Axios to connect with API which can use AXIOS to submit Get, Post, Put, Delete requests to the server. In Component App.js file, manage different pages and save the user's token status is possible.

with the desired codes added, the project currently has an API-connected booking page and a login page. After logging in and receiving a token, the user can view their server-based reservations on the Bookings page.

About style of site, a section of the design was created using HTML, CSS, and bootstrap. Bootstrap and other libraries were used to improve the user interface. The JWT token is saved locally (LocalStorage or Context) upon user login and provided to any further requests to access restricted routes (like /bookings) to preserve security and control access. It means, JWT tokens were used to show reservations (Bookings) to user.

After installing all required components, used this command to run react “npm start”. This command causes port 3000 to be used by the program. React automatically starts a local

development server, which is visible by launching a browser window at http://localhost: 3000. The browser will automatically navigate to http: // localhost: 3000 after the NPM Start command has been executed.

To collect user reservations from the back end, an API /Bookings request must be made first. This can be accomplished by viewing only the user bookings that are currently in effect using the JWT token that was given to the user upon arrival.
Add the token to the request header after that. For the server to recognize the user and only return the reservations, you must include the JWT token that the user obtains when they arrive in the Axios request header. On the other hand, it will display the message "You have no bookings Yet" if there are no reservations.

Totally, app.js is the initial file that controls other elements and routes using the React Router framework. Routes for bookings, registration, and login are included in this file.
Index.js: Your application's initial code that uses React Router to render the App Component in the Root element.
Header.js: Contains the application header with links to view further pages, log in, and exit the account.
Bookings.js: A page that uses the JWT token to display a user booking list that gets reservations from the server.
Login.js: The login page where users enter their email address and password to log in and obtain tokens. And the registration page that enables users to register a new account is called Signup.js.

8. Conclusion

This project combines modern web development technology with reliable database administration to meet the demand for a scalable and secure hotel and room management system. The PostgreSQL database offers dependable and structured data storage, while the backend server, which is built using Node.js and Express.js, effectively manages API calls. Using bcrypt for password hashing and JWT (JSON Web Tokens) for user session management, secure user authentication and role-based access control are accomplished, guaranteeing a secure and effective solution for both administrators and ordinary users. The project gives administrators the ability to easily generate, change, and monitor data through dynamic CRUD operations for managing hotels, rooms, and bookings. A responsive and user-friendly interface is provided via frontend development using React.js, allowing for smooth system interaction. While pgAdmin makes managing and debugging PostgreSQL data easier, Postman is used to test APIs for dependability and pgAdmin is used to verify database connectivity and data integrity. Applying Sequelize as the ORM, the project reduces the complexity of working with SQL queries by achieving a clear, manageable framework for database operations.

9. References

geeksforgeeks (2018). *Introduction to Postman for API Development*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/introduction-postman-api-development/.

GeeksforGeeks. (2019). *JSON web token | JWT*. [online] Available at: https://www.geeksforgeeks.org/json-web-token-jwt/.

MongoDB. (n.d.). *What Is Full Stack Development? | A Complete Guide*. [online] Available at: https://www.mongodb.com/resources/basics/full-stack-development.

Amazon Web Services (2024). *What is PostgreSQL? – Amazon Web Services*. [online] Amazon Web Services, Inc. Available at: https://aws.amazon.com/rds/postgresql/what-is-postgresql/.