OULUN AMMATTIKORKEAKOULU

Quynh Nguyen

THE HOTEL BOOKING APP

THE HOTEL BOOKING APP

Quynh Nguyen
Bachelor's thesis
Spring 2023
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Information Technology

Author(s): Quynh Nguyen
Title of the thesis: The Hotel Booking App
Thesis examiner(s): Teemu Korpela
Term and year of thesis completion: Spring 2023 Pages: e.g., 37

While people's living standards are improving, the demand for domestic and foreign travel is increasing, and the need to find hotels online is increasingly necessary. Accessing the hotel booking application to search for booking information will help customers book their rooms.

In this bachelor's thesis, I created an application called Hotel Booking App, it focuses on exploring the possibilities of the importance of admin such as managing user login and registration requirements, managing the hotel reservation system user including check-in and check-out date, and the total amount to be paid. My booking application is created on Typescript based on structure OOP, JavaScript, NodeJS, ReactJS, Postman for API testing, and PSQL database. I have used MUI and CSS, HTML for Front End.

In this project I have put the skills acquired during my studies to the test, and based on those skills, created an application that is usable in real-life. As a potential commercial project, if well developed, there will be many advantages for my next projects.

Keywords: OOP, JavaScript, NodeJS, React, Postman for API testing and PSQL database, MUI, CSS, HTML.

PREFACE

When I do the thesis, I want to thank Mr. Teemu Korpela as my instructor. He is very enthusiastic in his work, always enthusiastically helping and giving me useful advice to complete this graduate thesis. And I also want to thank Mrs. Jaana Raudaskoski for helping me correct the mistakes I needed to create a complete thesis. I hope this hotel booking application will help people's increasing travel demand.

Thanks to my mother and my father and my whole family who supported me while I was studying at school. I have a lot of support to study like interacting with modern software tools, a perfect environment to learn, explore and experience the techniques in my writing.

Finally, I feel lucky to be in this school, which has given me good friends and teachers who are always ready to help in the learning process. I firmly believe that these will help me in the future.

Oulu, 25.5.2023
Quynh Nguyen

CONTENTS

LIST OF ABBREVIATIONS

OOP	Object-oriented programming
API	Application Programming Interface
PSQL	PostgreSQL
MUI	Multilingual User Interface
CSS	Cascading Style Sheets
OTAs	Online Travel Agencies
HTML	Hypertext Markup Language
JSON	JavaScript Object Notation

1 INTRODUCTION

The Hotel Booking App is generally a user and service management system of the hotel. It is operated through the hotel's website. When using this application, after the customer made a reservation, all data will transfer to the data manager immediately.

When a user accesses the Hotel Booking App, it will ask the user to register when they do not have an account or log in when they already have an account to book a room. In the login section, it will ask the user to enter the email address and password. The admin can manage users by finding usernames based on the user's email address which will show check-in and check-out dates as well as the amount to be paid.

The Hotel Booking App is a convenient hotel reservation application for customers, easily creating comfortable services for users when logging into the hotel's website, because it allows customers to book rooms online through the hotel's website and provide hoteliers with hotel channel management and other resources to improve the customer's experience and increase online bookings.

2 THE HOTEL BOOKING ONLINE WEBSITE

2.1 Hotel Booking Online Process

After the users have visited the hotel's website, they can search for the hotel information that they want to find such as booking information, room prices, and hotel services.

To book a room, the users must create an account on the hotel's website. To create an account, an active email address and a password are required.

When choosing a room to reserve, users will also have to log in to the website through that account and select the type of room that they want to register. From the user's reservation request, the website system will filter the search term and return the available rooms with the content corresponding to the search term requested by the users.

Most booking applications manage and transmit data through the hotel's website [1], if the user finds the required room, they will click on the room they want to select, and the application will ask the user to log in if they already have an account or register if they do not have an account, they must fill in the form like the customer's first and last name (username), address, phone number, check-in date, check-out date and select reservation button.

Once the room has been booked, it will show the amount to be paid, the check-in date and check-out date, along with the email address of the user booking. After that, the application will return an invoice showing the amount to be paid (the user can pay on arrival at the hotel location or pay online) [1].

2.2 Hotel Booking Online Model

The Hotel Booking Online model is an easy way to understand examples and describe the steps of Hotel Booking Online [2].

It also includes the user and the process when the user using it, and how the data is connected to the admin. Below is the model which is showing the booking process and the relationship between the user and the admin as shown in Figure 1.

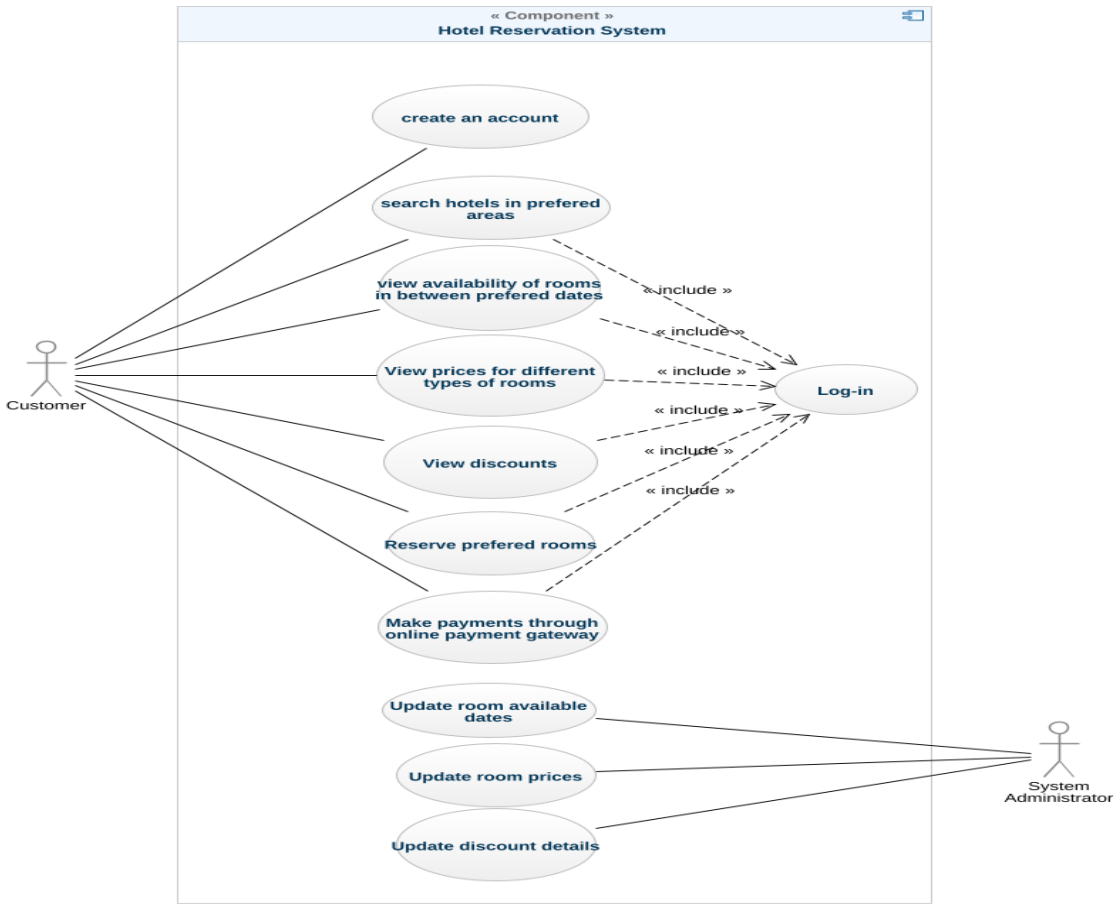

FIGURE 1. Hotel booking process use case diagram [2].

2.3 The advantages and disadvantages of the online hotel booking application

Nowadays, businesses are gradually making more business on the Internet. Booking hotel online is gradually being used by many hotels. Booking hotels online attracts more customers and saves human resources. However, booking hotel online also has advantages and disadvantages.

2.3.1 The advantages of the online hotel booking application

Online booking is often trusted by many people because it does not need to go to the hotel to make a reservation, they only need a computer, laptop, or phone with internet to make a reservation. While they are booking rooms, they also see the prices of the rooms, so they do not need to worry about the price, they can choose the price that suits their budget. They also search the hotel on Google, it will return a list of hotels and show them the reviews of each hotel. Booking Hotel Online is also very beneficial for people who want to travel for the first time.

- Booking Hotel Online will save more time [3]: when too many customers call, the staff will not be able to answer all the phones in the same amount of time, the phone line will be busy, and they will have to wait to call again. If on a holiday, it will need a lot of reservation staff to pick up the phone, so booking online is the best choice in that case and can save more human resources. Therefore, booking online will be a saviour instead of staff who always must be on call 24/24.

- Online booking helps to shorten the booking steps [3]: Customers will directly access the hotel website to make a reservation. They do not need to call the staff to ask about the services that the hotel has. All services are available on the hotel's website. They can search for what they want on the website by themselves.

- Booking Hotel Online will attract more customers [3]: If it is a small hotel, of course there are limitations for staff resources. Therefore, when many customers call at a time, the hotel staff cannot answer all of them and will lose customers' patience and customers will rate their rankings as a bad experience. So, to respond to the needs of customers, they can create their own hotel's website and users can book rooms right on the website without having to wait to call the reservation staff.

- On the hotel website, they can not only book rooms but also sell products that the hotel has [3]: On the hotel website, they can sell some products such as souvenirs, clothes, and the necessary products following the needs of customers. They can view the room's photos and user reviews on the website below that room's pictures.

- Booking Online will save customers who have booked [3]: Instead of having to create a digital book to save the customer's information, when customers click to book a room, the system will automatically store their information without having to write down and search for customer's information.

- The needs of customers are always at first [3]: Nowadays, there are more and more hotels are growing, thus causing increased competition between hotels. Therefore, they have to have a suitable price that matches the needs of the customers. And more than that, they also have to attract more customers by offering many promotions during the holidays or lowering the price to be reasonable.

2.3.2 The disadvantages of using an online hotel booking application

- You must have Wi-Fi or 4G [4]: When you want to make a reservation online, you must find yourself a computer or phone with the internet to book a hotel room as well as view your reservation information on the hotel page.

- Booking online will update more customers [4]: Many hoteliers make a lot of money and attract more customers. But if they are small business hotel owners, it will be a big challenge for them. Because the number of customers is overloaded, there are not enough rooms and the reservation staff is also lacking, requiring them to recruit more staff.

- Booking online websites are not always the same [4]: Nowadays many fraudulent hotels appear and lose more trust from users of online hotel booking websites. The room's pictures on the hotel website may not be the same as the real room in the hotel place, so customers should come and check the room directly.

3 BASIC TECHNOLOGY

3.1 Front-end

3.1.1 Material UI

Material UI is a CSS tool that supports user interface design [6]. It will assist in designing beautiful applications and save time for software developers instead of having to think about CSS.

3.1.2 ReactJS

React is a JavaScript framework for building user interfaces. It is run on a website. React was developed by a group of programmers at the Facebook company. It was introduced as an open-source JavaScript engine in 2013 [7].

3.1.3 HTML

HTML stands for Hypertext Markup Language. HTML empowers authors to effortlessly publish documents online, incorporating captivating titles, articulate text, intricate tables, comprehensive lists, and captivating images. Furthermore, it enables users to effortlessly retrieve information online through the seamless integration of hypertext links, ensuring that desired content is just a click away.

HTML document is made up of HTML elements defined by pairs of tags (tags and attributes) [8]

3.1.4 CSS

CSS (Cascading Style Sheets) is a language used to design websites, combined with HTML to create web pages. It can be easier to understand as follows: HTML could be seen as the skeleton of the system, and CSS is built on top of that basic structure.

3.1.5 JavaScript

JavaScript is the most popular programming language. [10] JavaScript allows us to do complex things on the web. [7] It is created to add interactivity to HTML. As we know above HTML is the skeleton of the human body, CSS is the skin and JavaScript is the muscles of the human body.

3.2 Back-end

3.2.1 TypeScript

TypeScript is an enhanced version of JavaScript; it was created to improve the functionality that basic JavaScript can hardly do. TypeScript has more functionality than JavaScript. TypeScript also can work widely for applications of Angular2 and Nodejs languages [11].

3.2.2 NodeJS

NodeJS is an extremely powerful JavaScript platform used to develop online chat apps, live videos, and web page apps.This platform was developed by Ryan Dahl in 2009 [12].

3.2.3 Express.js

Express.js is a framework for NodeJS [13]. Express is a small and utility framework for building web apps, providing many powerful features for developing web and mobile applications. It is very easy to develop Node.js-based agile applications for Web apps.

3.3 Database

3.3.1 PostgreSQL

PSPQL is a well-known old and very reliable database management system. PostgreSQL was created over 20 years ago.

One of the most obvious advantages of PostgreSQL is not having to pay any fees or royalties. Accordingly, you will be free to use, modify and distribute PostgreSQL in any form [14].

3.4 Testing

3.4.1 Postman

Postman the tools to test APIs. Postman was developed by Abhinav Asthana in 2012. It also allows to save the history of requests [15].

4 IMPLEMENTATION

Currently, the application has been built after coding and is running on the local host platform. The application is used to perform the main steps in customer management and management of reservations and locations worldwide.

The basic functions of the application include creating users and booking hotels with time, price, and location. Managers can search for specific customers based on email addresses. The application is connected between the interface website and the server side via TCP. All customer data will be saved in the local storage database and executed to read documents through the server.

4.1 Build a Restful API

4.1.1 Setting database

To implement the structure that connects the executed data to the database, it needs to be set up with specific information including the name of the data, the owner of the database, the port of the application, the user, and the password. Here typescript supports the “Sequelize” function for database connection.

Here are the main steps in how to connect to the database.

1. Setting NPM.
2. Install “Sequelize”.
3. Create a connection pool.

For the database to be complete and consistent, models are needed as table names in the database. After the settings are complete, the parts are written in the hidden form, this makes the user and the public source unaffected by 3rd parties. It will help the users to experience the application and connect directly to the database they want. To do this, dozens released by C# have improved the anonymization of unnecessary and set items that can be under the user-supplied .env file name as seen in Figure 5.

```
POSTGRES_HOST=localhost
POSTGRES_PORT=5432
POSTGRES_USER=postgres
POSTGRES_PASSWORD=tainguyen
POSTGRES_DB=booking_hotel
```

FIGURE 5. Setting environment for the database.

The last thing mentioned here is connection control. This helps us to detect error codes or successful connections to the database as shown in Figure 6.

```
import Logging from '../library/Logging';
import { User } from '../model/User';
import { Booking } from '../model/Booking';
dotenv.config();

class Database {
    public sequelize: Sequelize | undefined;

    private POSTGRES_DB = process.env.POSTGRES_DB as string;
    private POSTGRES_HOST = process.env.POSTGRES_HOST as string;
    private POSTGRES_PORT = process.env.POSTGRES_PORT as unknown as number;
    private POSTGRES_USER = process.env.POSTGRES_USER as unknown as string;
    private POSTGRES_PASSWORD = process.env.POSTGRES_PASSWORD as unknown as string;

    constructor() {
        this.connectToPostgreSQL();
    }

    private async connectToPostgreSQL() {
        this.sequelize = new Sequelize({
            database: this.POSTGRES_DB,
            username: this.POSTGRES_USER,
            password: this.POSTGRES_PASSWORD,
            host: this.POSTGRES_HOST,
            port: this.POSTGRES_PORT,
            dialect: 'postgres',
            models: [User, Booking]
        });

        await this.sequelize
            .authenticate()
            .then(() => {
                Logging.info('PostgreSQL Connection has been established successfully.');
            })
            .catch((err: any) => {
                Logging.error(`Unable to connect to the PostgreSQL database: ${err}`);
            });
    }
}

export default Database;
```

FIGURE 6. Connect to the database.

4.1.2 Model Database

After connecting to the database, now everything in the database seems to be empty and has not been structured anything. The next step is to create the schemas for the database and the columns we want to use in the application such as user, password, etc. To do this we will use the keywords used in the typescript as a tool. Helpful support saves us time and logic will be completed when called by the server as shown in Figures 7 and 8.

```
import { Model, Table, Column, DataType, BelongsTo, ForeignKey } from 'sequelize-typescript';
import { User } from './User';

@Table({
    tableName: Booking.BOOKING_TABLE_NAME
})
export class Booking extends Model {
    public static BOOKING_TABLE_NAME = 'booking' as string;
    public static BOOKING_ID = 'booking_id' as string;
    public static USER_ID = 'user_id' as string;
    public static VENUE = 'venue' as string;
    public static START_TIME = 'start_time' as string;
    public static END_TIME = 'end_time' as string;
    public static PRICE = 'price' as string;

    @Column({
        type: DataType.INTEGER,
        primaryKey: true,
        autoIncrement: true,
        field: Booking.BOOKING_ID,
        unique: true,
        allowNull: false
    })
    booking_id!: number;

    @ForeignKey(() => User)
    @Column({
        type: DataType.INTEGER,
        allowNull: false,
        references: {
            model: User,
            key: 'user_id'
        },
        field: Booking.USER_ID
    })
    user_id!: number;

    @BelongsTo(() => User)
    user!: User;
```

FIGURE 7. Booking Model

```
import { Model, Table, Column, DataType } from 'sequelize-typescript';

@Table({
    tableName: User.USER_TABLE_NAME
})
export class User extends Model {
    public static USER_TABLE_NAME = 'user' as string;
    public static USER_ID = 'user_id' as string;
    public static PASSWORD = 'password' as string;
    public static EMAIL = 'email' as string;

    @Column({
        type: DataType.INTEGER,
        primaryKey: true,
        autoIncrement: true,
        field: User.USER_ID,
        unique: true
    })
    user_id!: number;

    @Column({
        type: DataType.STRING(45),
        field: User.EMAIL,
        allowNull: false,
        unique: true
    })
    email!: string;

    @Column({
        type: DataType.STRING(45),
        field: User.PASSWORD,
        allowNull: false
    })
    password!: string;
}
```

FIGURE 8. User Model

4.1.3 Repository Database

Repository database is extremely important in performing data reading, data entry, data deletion, or data modification. We need the repository class to provide specific information and access to the database, the columns, and the rows in the database. We need to manage the error codes and check if the data is of the same type as shown in Figure 9.

```typescript
import { User } from '../model/User';

interface IUserRepo {
    retrieveAll(): Promise<User[]>;
    retrieveById(email: string): Promise<User>;
    save(user: User): Promise<void>;
}

export class UserRepo implements IUserRepo {
    async retrieveById(email: string): Promise<User> {
        try {
            const exist_user = await User.findOne({
                where: {
                    email: email
                }
            });
            if (!exist_user) {
                throw new Error('User not found!');
            }
            return exist_user;
        } catch {
            throw new Error('Failed to get a user!');
        }
    }
    async retrieveAll(): Promise<User[]> {
        try {
            return await User.findAll();
        } catch (err) {
            throw new Error('Failed to get all user!');
        }
    }
    async save(user: User): Promise<void> {
        try {
            await User.create({
                email: user.email,
                password: user.password
            });
        } catch (err) {
            throw new Error(`Failed to create a new user! ${err}`);
        }
    }
}
```

FIGURE 9. User Repository

4.1.4 Controller

MVC is a very popular model for structuring a web application, applied in many frameworks such as Spring, and dotNET core, its main component is the Controller. We can leave the request handling logic here because using typescript we will apply some OOP things as shown in Figure 10.

```
import { Request, Response, NextFunction } from 'express';
import HttpException from '../../library/exceptions/http.exception';
import { User } from '../../model/User';
import { UserRepo } from '../../repository/UserRepo';

class UserController {
    async findByEmail(req: Request, res: Response, next: NextFunction) {
        try {
            let email = req.params['email'];
            const new_user = await new UserRepo().retrieveById(email);

            res.status(200).json({
                status: 'Ok!',
                message: 'Successfully fetched a user data!',
                data: new_user
            });
        } catch (error: any) {
            next(new HttpException(400, error.message));
        }
    }

    async findAll(req: Request, res: Response, next: NextFunction) {
        try {
            const new_user = await new UserRepo().retrieveAll();

            res.status(200).json({
                status: 'Ok!',
                message: 'Successfully fetched all user data!',
                data: new_user
            });
        } catch (error: any) {
            next(new HttpException(400, error.message));
        }
    }
```

FIGURE 10. User Controller

4.1.5 Router

Route parameters are locations on the URL marked by naming, the purpose is to retrieve the corresponding values. All argument values will be placed on the request object in the params property. The attribute name matches the keyword defined on the URL as shown in Figure 11.

```
import BaseRouter from './base/BaseRouter';
import user from '../controller/user';
import validate from '../middleware/ validation.middleware';
import { createUserSchema } from '../schema';

class UserRouter extends BaseRouter {
    public routes(): void {
        this.router.post('', validate(createUserSchema), user.create);
        this.router.get('', user.findAll);
        this.router.get('/:email', user.findByEmail);
    }
}

export default new UserRouter().router;
```

FIGURE 11. User Router

4.1.6 Library

In writing code, the most important thing is to manage the information, warnings, and error codes that are appearing in the program. This is carefully set up so that management information can be captured on the server through “nodemon” and built-in libraries. The information will be updated specifically by date and time. This helps us manage code flexibly and specifically as shown in Figure 14.

```
import chalk from 'chalk';

class Logging {
    public static log = (args: any) => this.info(args);
    public static info = (args: any) => console.log(chalk.blue(`[${new Date().toLocaleString()}] [INFO]`), typeof args === 'string' ? chalk.blueBright(args) : args);
    public static warning = (args: any) => console.log(chalk.yellow(`[${new Date().toLocaleString()}] [WARN]`), typeof args === 'string' ? chalk.yellowBright(args) : args);
    public static error = (args: any) => console.log(chalk.red(`[${new Date().toLocaleString()}] [ERROR]`), typeof args === 'string' ? chalk.redBright(args) : args);
}

export default Logging;
```

FIGURE 14. Library Handler

4.1.7 Middleware

Middleware in software engineering is defined as a piece of software that acts as a bridge, providing services from the operating system to applications, and helping applications to interact with components. part allowed by the operating system.

1. Find the Route corresponding to the request.
2. Use CORS Middleware to check the request's cross-origin Resource sharing.
3. Use CRSF Middleware to authenticate the request's CSRF, anti-fake request.
4. Use Auth Middleware to verify whether the request is accessed or not.
5. Handle the work requested by the request (Main Task).

The code can be observed in detail in the middleware, below are Figures 15 and 16

```
import { Request, Response, NextFunction } from 'express';
import HttpException from '../library/exceptions/http.exception';

const errorMiddleware = (error: HttpException, req: Request, res: Response, _next: NextFunction): void => {
    const status = error.status || 500;
    const message = error.message || 'Something went wrong';
    res.status(status).send({
        status,
        message
    });
};

export default errorMiddleware;
```

FIGURE 15. Errors Middleware

```
import { NextFunction, Request, Response } from 'express';
import { AnyZodObject } from 'zod';

const validate = (schema: AnyZodObject) => async (req: Request, res: Response, next: NextFunction) => {
    try {
        await schema.parseAsync({
            body: req.body,
            query: req.query,
            params: req.params
        });

        return next();
    } catch (err: any) {
        const error_message = JSON.parse(err.message);
        return res.status(400).json({
            status: 'Bad Request!',
            message: error_message[0].message
        });
    }
};

export default validate;
```

FIGURE 16. Validation Middleware

4.1.8 Error Handling

An extremely essential part of the app is error handling, we will apply a middleware to handle this. Note: Since express runs middleware chained from start to finish, you must call "initializeErrorHandling" last.

After configuring the exception, now we apply it to the hero controller, when getting hero by ID but not found will return an exception named "HeroNotFoundException" as shown in Figure 17.

```
class HttpException extends Error {
    public status: number;
    public message: string;

    constructor(status: number, message: string) {
        super(message);
        this.status = status;
        this.message = message;
    }
}

export default HttpException;
```

FIGURE 17. HTTP Exception Handler

4.2 Web client interface

To design every interface for the website we need to pay attention to the colors, layout, and functions that we want to aim for. Here we are only interested in the shortcut interface so that the admin can observe and manage it effectively and easily.

4.2.1 Components

Components React is a JavaScript library used to set up user interfaces. Components in React help to divide the user interface (UI) into smaller components for simple management and reuse. Each Component will take care of different display parts of the interface as shown in Figure 18.

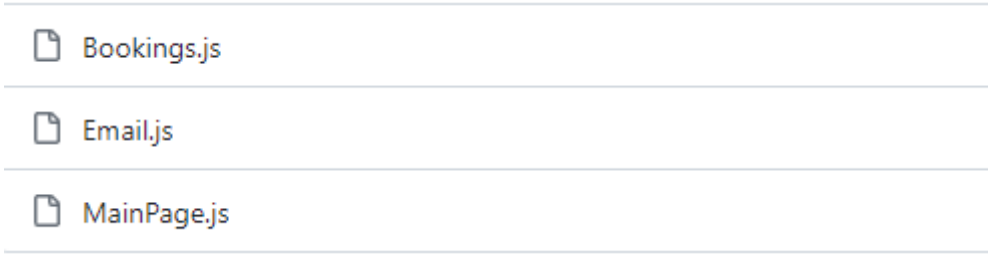

FIGURE 18. Component's structure

4.2.2 Store

The store stores the application state. You can access saved states, update, subscribe, or unsubscribe to listeners. The actions taken will return to the new state. So, this state is shown as shown in Figure 19.

```
import { combineReducers } from "redux";
import { configureStore } from "@reduxjs/toolkit";
import bookingReducer from '../features/booking/bookingSlice';


const reducer = combineReducers({
    bookings: bookingReducer,
});

export default configureStore({
    reducer,
    middleware: getDefaultMiddleware =>
        getDefaultMiddleware({
            serializableCheck: false,
        })
});
```

FIGURE 19. Store High Level

4.2.3 Reducers

The reducer will take the state, execute it, and return the new state to the application. The states are saved as objects, and they will indicate the change of states to go to the action for the store as shown in Figure 20.

```
export const bookingSlice = createSlice({
    name: 'booking',
    initialState,
    reducers: {
        reset: (state) => initialState
    },
    extraReducers: (builder) => {
        builder
            .addCase(getAllBooking.pending, (state) => {
                state.isLoading = true
            })
            .addCase(getAllBooking.fulfilled, (state, action) => {
                state.isLoading = false
                state.isSuccess = true
                state.bookings = action.payload
            })
            .addCase(getAllBooking.rejected, (state, action) => {
                state.isLoading = false
                state.isError = true
                state.message = action.payload
            })
            .addCase(getUserBooking.pending, (state) => {
                state.isLoading = true
            })
            .addCase(getUserBooking.fulfilled, (state, action) => {
                state.isLoading = false
                state.isSuccess = true
                state.bookings = action.payload
            })
            .addCase(getUserBooking.rejected, (state, action) => {
                state.isLoading = false
                state.isError = true
                state.message = action.payload
            })
    }
})
```

FIGURE 20. Reducers Execution

5 TESTING

Testing API is crucial in the software development process for several reasons:

**Ensuring Functionality: ** API testing helps verify APIs. By systematically testing different API endpoints and scenarios, you can identify any functional issues or bugs early on, ensuring the API behaves as intended.

**Validating Integration: ** APIs are often used to integrate various software components or systems. Testing the API helps validate the integration between these components, ensuring they can communicate and exchange data accurately. It allows you to verify that data is transmitted correctly, error handling is implemented appropriately, and compatibility across different systems is maintained.

**Detecting Issues and Bugs: ** API testing helps uncover issues such as incorrect data formats, missing parameters, authentication failures, or inconsistent responses. Detecting and addressing these problems early in the development lifecycle reduces the risk of encountering critical errors in production environments.

**Ensuring Reliability and Stability: ** Thorough API testing contributes to the reliability and stability of the overall software system. By validating the API's functionality and behavior, you can increase confidence in its reliability, reducing the chances of unexpected failures or system downtime.

**Promoting Security: ** APIs often handle sensitive data or perform critical operations, making security testing essential. By thoroughly testing the API for vulnerabilities, such as injection attacks or unauthorized access attempts, you can identify and address security weaknesses, protecting the system and the data it handles.

**Facilitating Collaboration: ** APIs are commonly used by multiple developers or teams working on different components of a project. Properly tested APIs provide clear documentation, defined inputs and outputs, and predictable behavior, making collaboration between teams smoother and more efficient.

5.1 Create Users

To properly explain the process of using Postman with the "create user" function and the "body-parse JSON" type, we can break it down into a few steps:

1. Start by opening Postman and creating a new request. Make sure you have the necessary end-point URL for the "create user" function.

2. Within the request, specify the HTTP method as "POST" since you are creating a new user.

3. Next, navigate to the "Body" tab in Postman. Here, you can select the "raw" option and choose the data type as "JSON" from the dropdown menu.

4. In the request body, you need to provide the necessary details to create a user. Typically, this involves including relevant key-value pairs in JSON format as shown in Figure 21:

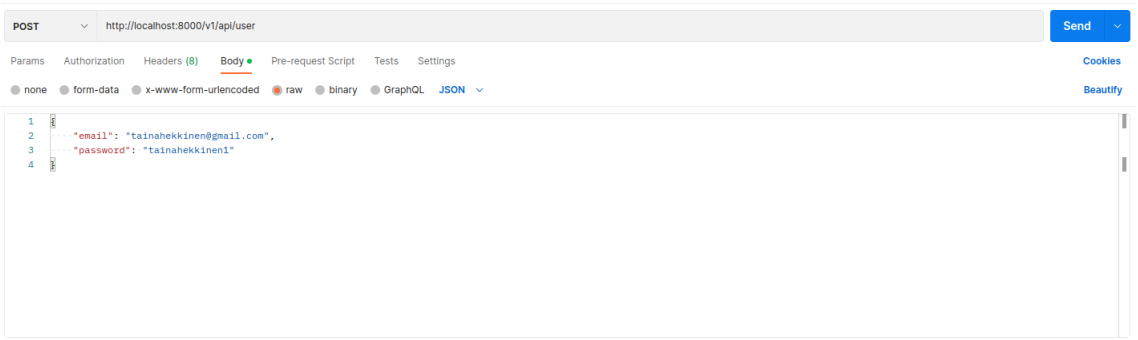

FIGURE 21. Testing Create Users

5. Once you have entered the required information, you can send the request by clicking on the "Send" button in Postman.

6. The server will process the request and respond with the appropriate result. You can check the response in the "Response" section of Postman.

By following these steps and utilizing the "body-parse JSON" type in Postman, you can effectively test the "create user" function and ensure that the server properly receives and processes the JSON data you provide.

5.2 Booking Hotels

To create a booking for a room based on the IDs of users using Postman, you can follow these steps:

1. Open Postman and create a new request. Make sure you have the endpoint URL for creating a booking.

2. Set the HTTP method to "POST" since you're creating a new booking.
3. In the request body, specify the necessary details for the booking, including the room ID and the user IDs involved. You can structure the JSON body as shown in Figure 22:

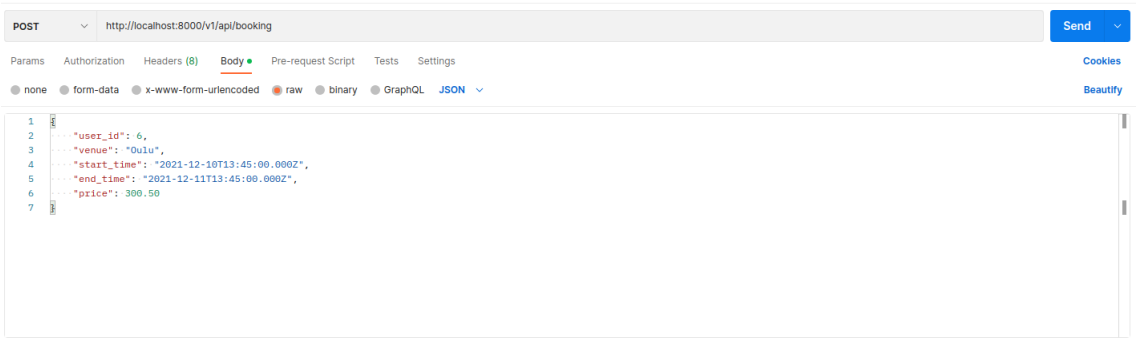

FIGURE 22. Testing Reserve Hotel

Replace "123456" with the actual ID of the room you want to book. Modify the "user_ids" array to include the relevant IDs of the users participating in the booking.
4. Once you've entered the required information, click on the "Send" button in Postman to submit the request.
5. The server will process the request and respond with the result. You can check the response in the "Response" section of Postman to verify if the booking was successfully created.
By following these steps, you can utilize Postman to create a booking for a room based on the provided user IDs.

6 ADMIN VIEW

To ensure compatibility across different devices such as mobile and PC, it is important to create a responsive table that displays all the necessary information of the customer. By utilizing responsive design techniques, the table can adapt and adjust its layout based on the screen size and resolution of the device being used. This allows users to conveniently access and view customer information regardless of the device they are using, providing a consistent and user-friendly experience.

The table designed to display customer information can be enhanced with the functionality to sort data from any column. By incorporating sorting capabilities, users can easily organize and arrange the information based on their preferences.

This feature allows users to click on the column headers to sort the table in ascending or descending order based on the values in that specific column.

By enabling sorting functionality, the table provides a flexible and interactive way for users to explore and analyze customer data based on different criteria. It enhances usability and enables users to quickly find the desired information more efficiently.

To facilitate searching for a user with a specific email, you can incorporate a search button within the table of customer information. This search function allows users to input an email address and retrieve the corresponding user's details. Here is how it can be implemented:

1. Display a search input field and a search button above or adjacent to the table of customer information.
2. When a user enters an email address into the search input field and clicks the search button, triggers a search function.
3. The search function should iterate through the table rows, comparing the entered email with the email column in each row.
4. If a match is found, highlight the corresponding row or display only the matching row in the table.
5. Additionally, you can provide a feedback message indicating whether a user with the specified email was found or not.

By incorporating this search functionality with a search button, users can easily find and retrieve the information of a specific user based on their email address from the table of customer information as shown in Figure 23.

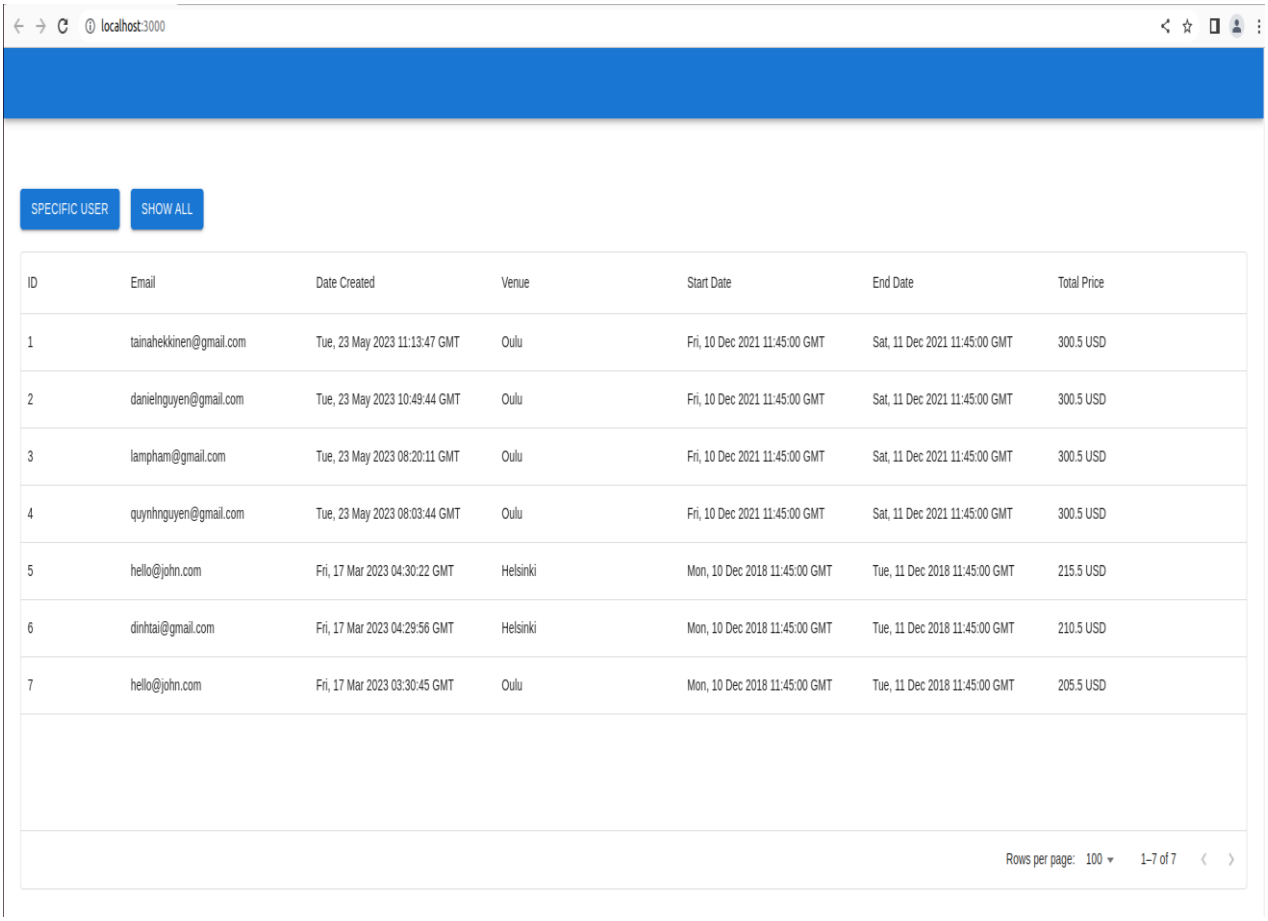

FIGURE 23. Customer Information Sheet

7 CONCLUSION

Reflecting back to the benefits and skills gained from the thesis, it is important to acknowledge the long-term impact the completion of this project could have on my future career. The use of ReactJS and Node.js in my thesis not only demonstrates my proficiency in these technologies but also makes me a competitive candidate for future career opportunities.

I have completed this thesis incorporating these on-demand technologies, and I have demonstrated my ability to self-study and take advantage of modern tools. This achievement not only validates my skills but also highlights my ability to tackle complex problems and come up with powerful solutions.

The profession of software development is increasingly valued and used by companies because they are widely applied and have many advantages that they bring in the process of web development. As a result, employers in various industries are increasingly looking for professionals with expertise in ReactJS and Node.js. My thesis, which demonstrates my ability to develop scalable and efficient applications using these technologies, positions me as an asset to organizations looking to take advantage of these technologies. use ReactJS and Node.js for their projects.

In addition to the capabilities that I have achieved during the completion of the project, I also have challenges to face while working on the thesis I also have some difficulties in the back end because I specialize in the front end. Applying Typescript programming language was also a turning point for me because I have not been exposed to it and have not had time to apply it. Being a parent created some challenges with the planning, and my adaptability and scheduling skills proved to be useful in this situation. These experiences, and hardships, along with my technical background, enhance my overall profile and increase my ability to work in the job market.

In addition, the thesis serves as a tangible and important project that can be featured in my portfolio or job interviews. Real-world implementation of ReactJS and Node.js in a real-world situation demonstrates my ability to apply theoretical knowledge and deliver tangible results.

In summary, the successful completion of my thesis combining ReactJS and Node.js not only demonstrates my technical expertise but also enhances my potential to secure a new career in the future. Using these technologies demonstrates my adaptability, problem-solving skills, and ability to deliver efficient web applications. By leveraging the experience gained and leveraging my thesis project, I can position myself as a highly sought-after professional in the industry, opening exciting career opportunities.

REFERENCES

[1] Pamela Salon, 2021. How Does an Online Hotel Reservation System Work? Date of retrieval 26.5.2023

https://www.linkedin.com/pulse/how-does-online-hotel-reservation-system-work-pamela-salon

[2] Nym, 2022. Use Case Diagram for Online Hotel Reservation System. Date of retrieval 26.5.2023

https://itsourcecode.com/uml/use-case-diagram-for-hotel-reservation-system/?utm_content=expand_article

[3] Amrigovoyage. What Are the Advantages of An Online Hotel Booking System? Date of retrieval 26.5.2023

https://amerigovoyage.com/blog/advantages-of-online-hotel-booking-system

[4] Foreupgolf. Advantages and Disadvantages of Using an Online Booking System. Date of retrieval 26.5.2023

foreupgolf.com/blog/advantages-and-disadvantages-of-using-an-online-booking-system/

[5] Matthieu Mauguin, 2021. What Is Next for Hotel Online Bookings? Date of retrieval 26.5.2023

https://www.hospitalitynet.org/opinion/4104231.html#void

[6] Material-UI: A popular React UI framework. Official documentation. Date of retrieval 27.5.2023

https://material-ui.com/

[7] ReactJS: A JavaScript library for building user interfaces. Official documentation. Date of retrieval 27.5.2023

https://reactjs.org/

[8] HTML (Hypertext Markup Language): The standard markup language for creating web pages. W3C Recommendation Date of retrieval 27.5.2023

https://www.w3.org/TR/html52/

[9] CSS (Cascading Style Sheets): The style sheet language used for describing the presentation of a document written in HTML. W3C Recommendation. Date of retrieval 29.5.2023

https://www.w3.org/Style/CSS/Overview.en.html

[10] JavaScript: A high-level programming language used for creating interactive and dynamic web content. ECMAScript 2021 Language Specification. Date of retrieval 29.5.2023

https://tc39.es/ecma262/2021/

[11] TypeScript: A typed superset of JavaScript that compiles to plain JavaScript. Official documentation Date of retrieval 28.5.2023

https://www.typescriptlang.org/

[12] Node.js: A JavaScript runtime built on Chrome's V8 JavaScript engine, used for server-side and networking applications. Official documentation. Date of retrieval 15.5.2023

https://nodejs.org/

[13] Express.js: A fast and minimalist web application framework for Node.js. Official documentation. Date of retrieval 20.5.2023

https://expressjs.com/

[14] PostgreSQL: An open-source relational database management system. Official documentation. Date of retrieval 22.5.2023

https://www.postgresql.org/

[15] Postman: A popular collaboration platform for API development and testing. Official documentation. Date of retrieval 24.5.2023

https://www.postman.com/