AGH – UNIVERSITY OF SCIENCE AND TECHNOLOGY

Project documentation for

Railway Reservation System

Object-oriented programming languages

Electronics and Telecommunication EN, III year

Kamil Kras

lecturer: Rafał Frączek

28.01.2025r.

1. Project description

The Railway Reservation System is designed to manage train bookings, user registrations, reservations, cancellations, and ticket generation. It integrates multiple components such as users, trains, tickets, and reservations to provide a seamless experience for passengers..

2. User's manual

How the System Works

1. **Main Menu (main.cpp):**
   - Displays options: Register, Login, or Exit.
   - Directs users to submenus based on their input.
2. **Logged-in Menu:**
   - Displays options:
     - View available trains
     - Book a ticket
     - View reservations
     - Cancel a reservation
     - View tickets
     - Logout
   - Handles corresponding operations using other classes.
3. **Data Persistence:**
   - User data is stored in `users.json`.
   - Train details are stored in `trains.json`.
   - Tickets and reservations are stored in `tickets.json`.

3. Compilation

Compilation Process

Steps to Compile and Run the Project:

1. **Prerequisites:** Ensure that a C++ compiler (e.g., g++) is installed and properly configured.
2. **Compile Command:**
   - Run the following command in the terminal to compile all the files:

```
g++ main.cpp ReservationManager.cpp TrainManager.cpp ticket.cpp
User.cpp train.cpp -o RailwayReservation.
```

3. **Run the Executable:**
   - After successful compilation, run the following command to start the application in :cmd: or in terminal

```
RailwayReservation.exe
```

```
./RailwayReservation
```

4. **File Structure:** Ensure the `Data` folder (containing `users.json`, `trains.json`, and `tickets.json`) exists in the same directory as the executable.
5. **Troubleshooting:**
   - Ensure all paths and dependencies are correct.
   - Verify that JSON files have the correct structure and are not empty.

4. Source files

Project Stucture

Header Files:

1. ***User.h*** *- Defines the User class and manages user information and operations.*
2. ***Train.h*** *- Defines the Train class, representing train details.*
3. ***ReservationManager.h*** *- Handles reservation operations such as booking and cancellation.*
4. ***TrainManager.h*** *- Manages train data, including loading and displaying trains.*
5. ***Ticket.h*** *- Facilitates ticket saving and searching.*

Implementation Files (.cpp):

1. **main.cpp** - Entry point of the application. Implements the main menu and user interaction.
2. **ReservationManager.cpp** - Implements reservation-related operations.
3. **TrainManager.cpp** - Implements train-related operations such as loading and displaying data.
4. **ticket.cpp** - Handles ticket management (saving and searching tickets).
5. **User.cpp** - Implements user-related operations like registration and login.

Data Files:

users.json *- Stores registered user information.*

trains.json *- Stores train details.*

tickets.json *- Stores reservation data.*

.

5. Dependencies

The following external libraries are used in the project:

- JSON for Modern C++ (json.hpp) A library for handling JSON data in C++. Website: https://json.nlohmann.me.

6. Class description

In the project the following classes were created:

1. User

Represents a user in the system.

- **Public Methods:**
- `int` **`getUserID()`** `const: Returns the user ID.`
- `std::string` **`getEmail()`** `const: Returns the email address of the user.`
- `std::string` **`getUsername()`** `const: Returns the username.`
- `std::string` **`getPassword()`** `const: Returns the user's password.`
- `bool` **`verifyPassword(const std::string& inputPassword)`** `const: Verifies the provided password against the user's password.`

- void **registerUser(std::vector<User>& users)**: Registers a new user, checking for duplicate emails.
- static void **saveToFile(const std::vector<User>& users)**: Saves the list of users to a file.
- static std::vector<User> **loadFromFile()**: Loads users from a file.
- static User* **loginUser(std::vector<User>& users, const std::string& email, const std::string& password)**: Handles user login by verifying email and password.
- void **addReservation(int trainID)**: Adds a reservation for the user.
- void **cancelReservation(int trainID)**: Cancels a reservation for the user.
- std::vector<int> **getReservations()** const: Returns a list of train IDs reserved by the user.
- std::string **getName()** const: Returns the name of the user.

---

2. Train

Represents a train in the system.

Public Methods:

- int **getTrainID()** const: Returns the unique ID of the train.
- std::string **getTrainName()** const: Returns the name of the train.
- std::string **getStartCity()** const: Returns the start city of the train's route.
- std::string **getEndCity()** const: Returns the destination city of the train's route.
- int **getAvailableSeats()** const: Returns the number of available seats.
- double **getTicketPrice()** const: Returns the price of a ticket for the train.
- std::string **getArrivalTime()** const: Returns the train's arrival time.
- std::string **getDepartureTime()** const: Returns the train's departure time.
- bool **reserveSeat()**: Reserves a seat on the train, reducing available seats by one.
- void **cancelSeat()**: Cancels a reserved seat, increasing available seats by one.
- void **setAvailableSeats(int seats)**: Updates the available seats for the train.

3. TrainManager

Manages train-related data and operations.

- **Public Methods:**
- static std::vector<Train> **loadTrains():** Loads trains from a JSON file.
- static void **updateAvailableSeats(std::vector<Train>& trains):** Updates the available seat count for trains based on reservations.
- static void **displayTrains(const std::vector<Train>& trains):** Displays the details of all trains.

---

4. ReservationManager

Handles reservations and ticket-related operations.

- **Public Methods:**
- void **setLoginStatus(bool status):** Updates the login status of a user.
- static void **displayUserReservations(const User& user, const std::vector<Train>& trains):** Displays the user's current reservations.
- void **cancelReservation(User& user, int trainID, std::vector<Train>& trains):** Cancels a user's reservation and updates the train's seat count.
- static void **viewTicket(const User& user, const std::vector<Train>& trains):** Displays ticket information for the user's reservations.

---

5. Ticket

Represents a ticket for a reservation

Public Methods:

- void **saveTicketToFile(const User& user, const Train& train):** Saves ticket information to a JSON file
- json **searchTicket(const std::string& username, int trainID)**

7. Resources

1. tickets.json

Description: A JSON file storing ticket details for users.

Structure:

- "arrivalTime": The arrival time of the train (e.g., "09:00 PM").
- "departureTime": The departure time of the train (e.g., "05:00 PM").
- "endCity": The destination city of the train (e.g., "Rzeszów").
- "startCity": The starting city of the train (e.g., "Lublin").
- "ticketPrice": The price of the ticket (e.g., 40.0).
- "trainID": The unique ID of the train (e.g., 5).
- "trainName": The name of the train (e.g., "Intercity E").

- "username": The username of the user who booked the ticket (e.g., "admin").

2. trains.json

Description: A JSON file storing information about all available trains.

Structure:

- "trainID": The unique ID of the train (e.g., 1).
- "name": The name of the train (e.g., "Express A").
- "startCity": The starting city of the train (e.g., "Kraków").
- "endCity": The destination city of the train (e.g., "Warszawa").
- "availableSeats": The number of available seats on the train (e.g., 100).
- "ticketPrice": The price per ticket for the train (e.g., 50.0).
- "arrivalTime": The arrival time of the train (e.g., "12:45 PM").
- "departureTime": The departure time of the train (e.g., "08:15 AM").

3. users.json

Description: A JSON file storing information about registered users.

Structure:

- "email": The email address of the user (e.g., "admin").
- "password": The password for the user account (e.g., "admin").
- "userID": The unique ID of the user (e.g., 1).
- "username": The username of the user (e.g., "admin").

These JSON files act as the primary data storage for the project, enabling user management, train information retrieval, and ticket booking functionality..

8. Future development

The project can be enhanced with the following features:

-API implementation. Implement APIs to fetch real-time train schedules, ticket availability, and delays directly from official train service providers.

- Develop a dedicated **Admin Mode** for managing the application's backend operations.

- Implementing a **Graphical User Interface (GUI)** for a more intuitive and user-friendly experience.

- Implement dynamic ticket pricing based on demand, time of booking, and seat availability.

- Payment Gateway Integration

- Possibility of adding function for Seat Selection

9. Other

I really enjoyed working on the Railway Reservation System. It has been a great opportunity to further develop my understanding of the key principles behind train ticket booking systems. There are still many additional features and functionalities that can be implemented in the future, offering plenty of room for further growth and improvements.