Glider 8 - Project Requirements Documentation

Project Overview

The "Glider 8" project is an airline reservation system designed to streamline the booking and management of flights. It supports distinct customer and admin roles, providing tailored functionalities for each.

Functional Requirements

User Roles

1. **Customer**

* Register an account with detailed personal information.
* Login with username and password.
* Retrieve forgotten passwords using a security question.
* Search for flights based on criteria (e.g., departure city, destination, date).
* Book flights and manage their reservations.
* Delete booked flights.
* Logout securely.

1. **Admin**

* Login with admin credentials.
* Manage customer reservations.
* Add, update, and delete flight records.
* Logout securely.

User Interface

Splash Screen

* Welcome users with a splash screen at application start.

Main Menu

* Options for customer registration and login.

Registration Form

* Input fields for first name, last name, address, ZIP code, state, username, password, email, SSN, and security question.

Login Form

* Username and password fields with an option to recover the password using a security question.

Customer Interface

* Search for flights.
* Book and delete flights.
* View booked flights.
* Logout functionality.

Admin Interface

All customer functionalities.

Additional capabilities to add, update, and delete flights.

Logout functionality.

Database Design

User Table

* username: Unique identifier for each use (primary key).
* password: Securely hashed passwords.
* role: Specifies if the user is a customer or admin.
* Other fields: first\_name, last\_name, address, zip, state, email, ssn, security\_question, security\_answer.

Flights Table

* flight\_number: Unique flight identifier (primary key).
* departure\_city, departure\_time: Origin details.
* destination\_city, destination\_time: Destination details.
* airline: Name of the airline.
* available\_seats: Number of unbooked seats.
* capacity: Total seat capacity.

Reservations Table

* reservation\_id: Unique reservation identifier (primary key).
* username: Associated customer.
* flight\_number: Linked flight details.

Key Features

Login

* Validate credentials and redirect based on user role.

Register

* Collect and store user information with validation.

Retrieve Password

* Validate the security question to recover the password.

Search Flights

* Filter flights based on user-provided criteria.

Book Flight

* Update reservations and flight availability.
* Prevent overbooking and duplicate reservations.

Delete Booked Flights

* Remove reservations and update seat availability.

Manage Flights (Admin)

* Add, update, and delete flight records in the database.

Non-Functional Requirements

Security

* Use secure methods to store passwords (e.g., hashing).
* Secure database connections.

Usability

* Ensure a user-friendly and intuitive interface.
* Optimize for responsiveness across devices.

Performance

* Ensure fast response times for queries and actions.
* Optimize database performance for scalability.

Reliability

* Implement robust error handling with user-friendly messages.
* Maintain high system uptime.

Technical Requirements

Programming Language: Java

Frameworks and Libraries

* JavaFX for the graphical user interface.
* JDBC for database interactions.

Database**:** MySQL

Build Tool**:** Maven

IDE**:** IntelliJ IDEA 2024.2.4

Development Environment:

Operating System: Windows + MacOS

Version Control**:** GitHub

Hosting**:** Azure Database for MySQL

Additional Considerations

Exception Handling and Logging

* Implement comprehensive exception handling for predictable errors.
* Log critical events for auditing and debugging purposes.

Coding Standards

* Follow industry best practices and conventions.

Testing

* Develop unit tests for core functionalities.
* Conduct end-to-end testing to ensure reliability.

References to Source Code

AdminPageController: Handles admin functionalities such as adding, updating, and deleting flight

BookingController: Manages flight search and booking operations for customers

DatabaseConnection: Establishes and maintains secure database connections

Flight Class: Represents flight data and its attributes

ForgotPasswordController: Enables password recovery via security questions

HelloApplication: Initializes the application and manages scene transitions

HelloController: Handles login and registration interactions

RegisterController: Facilitates new user registration with input validation

Reservations and ReservationsController: Manages flight reservations and their display to the user

UMLS

1. HelloController:

A screen shot of a computer

Description automatically generated

2. Booking Controller

+-------------------------------------+

| BookingController |

+-------------------------------------+

| - fromCityField: TextField |

| - toCityField: TextField |

| - datePicker: DatePicker |

| - timeField: TextField |

| - resultsTable: TableView<Flight> |

| - flightNumberColumn: TableColumn |

| - fromCityColumn: TableColumn |

| - toCityColumn: TableColumn |

| - dateColumn: TableColumn |

| - timeColumn: TableColumn |

| - seatsColumn: TableColumn |

| - searchButton: Button |

| - bookButton: Button |

| - backToLoginButton: Button |

| - backToReservationsButton: Button |

| - cancelSelectedFlightButton: Button|

| - bookedFlightsTable: TableView<Reservations> |

| - connection: Connection |

+-------------------------------------+

| + initialize(): void |

| + searchFlights(): void |

| + bookFlightButtonClick(): void |

| + backToLoginClick(): void |

| + navigateToReservations(event: ActionEvent): void |

| + handleCancelFlightButton(event: ActionEvent): void|

| - connectToDatabase(): void |

| - showAlert(alertType: AlertType, title: String, content: String): void |

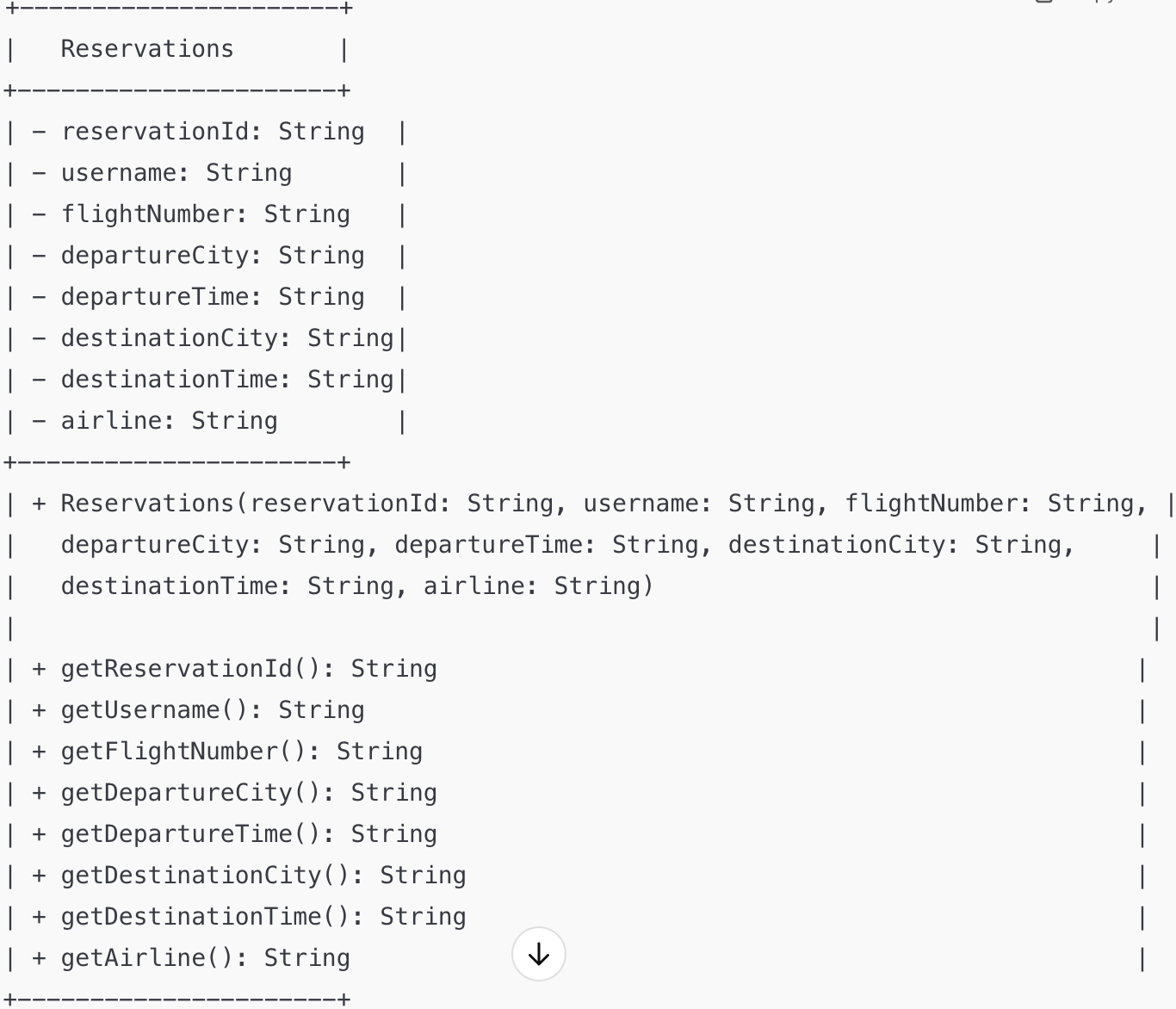
+-------------------------------------+

3. ForgotPasswordController :

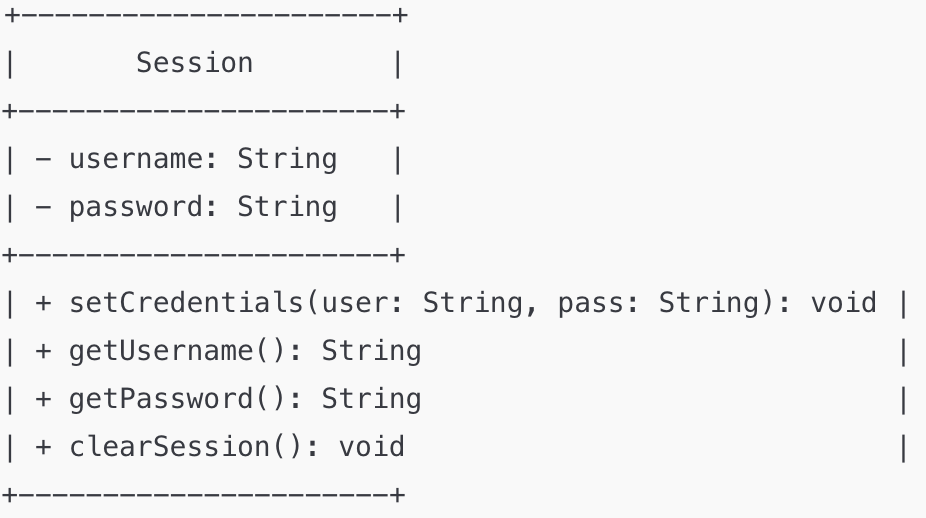


4. ReservationsController

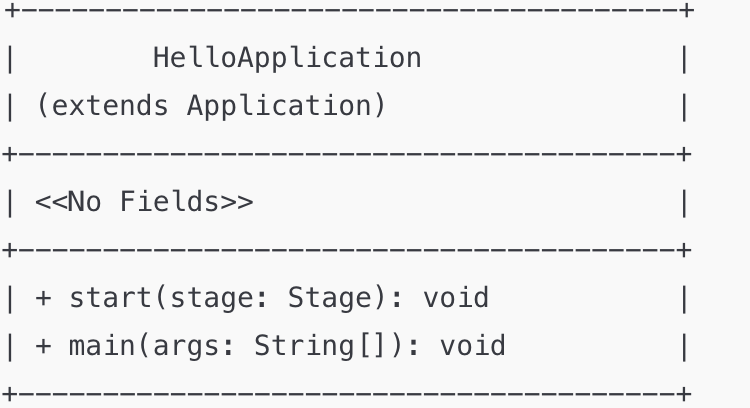


5. Reservation

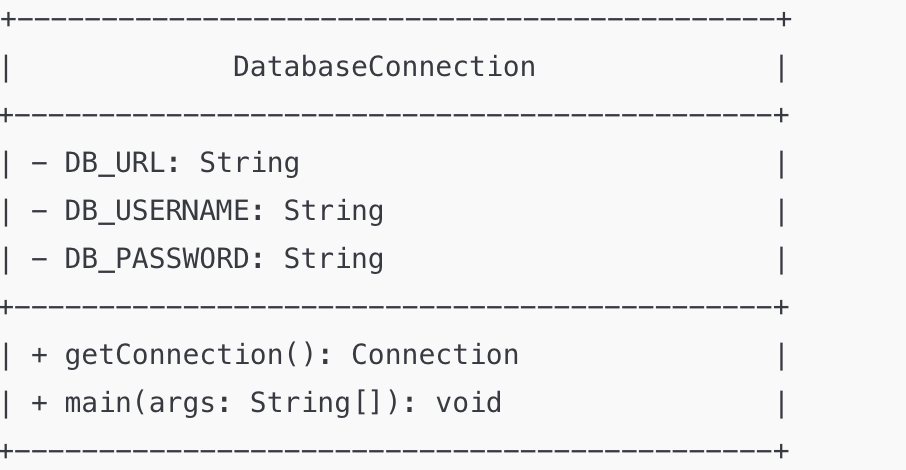
6. Sessions



7. Application



8. DatabaseConnection



9. AdminPageController

+------------------------------------------------------+

| AdminPageController |

+------------------------------------------------------+

| - flightNumberField: TextField |

| - departureCityField: TextField |

| - departureTimeField: TextField |

| - destinationCityField: TextField |

| - destinationTimeField: TextField |

| - airlineField: TextField |

| - availableSeatsField: TextField |

| - capacityField: TextField |

| - addFlightButton: Button |

| - allFlights: Button |

| - cancelFlightButton: Button |

| - updateFlightButton: Button |

| - backToLoginButton: Button |

| - bookedFlightsTable: TableView<Flight> |

| - flightNumberColumn: TableColumn<Flight, String> |

| - departureCityColumn: TableColumn<Flight, String> |

| - departureTimeColumn: TableColumn<Flight, String> |

| - destinationCityColumn: TableColumn<Flight, String> |

| - destinationTimeColumn: TableColumn<Flight, String> |

| - airlineColumn: TableColumn<Flight, String> |

| - availableSeatsColumn: TableColumn<Flight, String> |

| - capacityColumn: TableColumn<Flight, String> |

| - flightList: ObservableList<Flight> |

| - connection: Connection |

+------------------------------------------------------+

| + initialize(): void |

| - connectToDatabase(): void |

| - fetchAllFlights(): void |

| + addFlight(): void |

| + updateFlight(): void |

| + deleteFlight(): void |

| + backToLoginButtonClick(event: ActionEvent): void |

| - showAlert(alertType: AlertType, title: String, |

| content: String): void |

| + viewAllFlights(): void |

| + updateFlightAttribute(flightNumber: String, |

| column: String, newValue: String): void |

| + handleUpdateButtonClick(): void |

| + backToLoginClick(): void |

+------------------------------------------------------+

10. Flight

+-------------------------------------------------------+

| Flight |

+-------------------------------------------------------+

| - flightNumber: String |

| - departureCity: String |

| - departureTime: String |

| - destinationCity: String |

| - destinationTime: String |

| - airline: String |

| - availableSeats: String |

| - capacity: String |

+-------------------------------------------------------+

| + Flight(flightNumber: String, departureCity: String, |

| departureTime: String, destinationCity: String, |

| destinationTime: String, airline: String, |

| availableSeats: String, capacity: String) |

+-------------------------------------------------------+

| + getFlightNumber(): String |

| + getDepartureCity(): String |

| + getDepartureTime(): String |

| + getDestinationCity(): String |

| + getDestinationTime(): String |

| + getAirline(): String |

| + getAvailableSeats(): String |

| + getCapacity(): String |

| + setFlightNumber(flightNumber: String): void |

| + setDepartureCity(departureCity: String): void |

| + setDepartureTime(departureTime: String): void |

| + setDestinationCity(destinationCity: String): void |

| + setDestinationTime(destinationTime: String): void |

| + setAirline(airline: String): void |

| + setAvailableSeats(availableSeats: String): void |

| + setCapacity(capacity: String): void |

+-------------------------------------------------------+

11. User

+-------------------------------------------------------------+

| User |

+-------------------------------------------------------------+

| - username: String |

| - password: String |

| - firstName: String |

| - lastName: String |

| - email: String |

| - address: String |

| - state: String |

| - city: String |

| - zipcode: String |

| - securityQuestion: String |

| - securityAnswer: String |

| - role: String = "Customer" |

| - flights: ArrayList<Flights> |

| - action: String |

+-------------------------------------------------------------+

| + User() |

| + User(username: String, password: String, firstName: String,|

| lastName: String, email: String, address: String, |

| city: String, state: String, zipcode: String, |

| securityQuestion: String, securityAnswer: String) |

+-------------------------------------------------------------+

| + getUsername(): String |

| + setUsername(username: String): void |

| + getFirstName(): String |

| + setFirstName(firstName: String): void |

| + getLastName(): String |

| + setLastName(lastName: String): void |

| + getPassword(): String |

| + setPassword(password: String): void |

| + getEmail(): String |

| + setEmail(email: String): void |

| + getFlights(): ArrayList<Flights> |

| + setFlights(flights: ArrayList<Flights>): void |

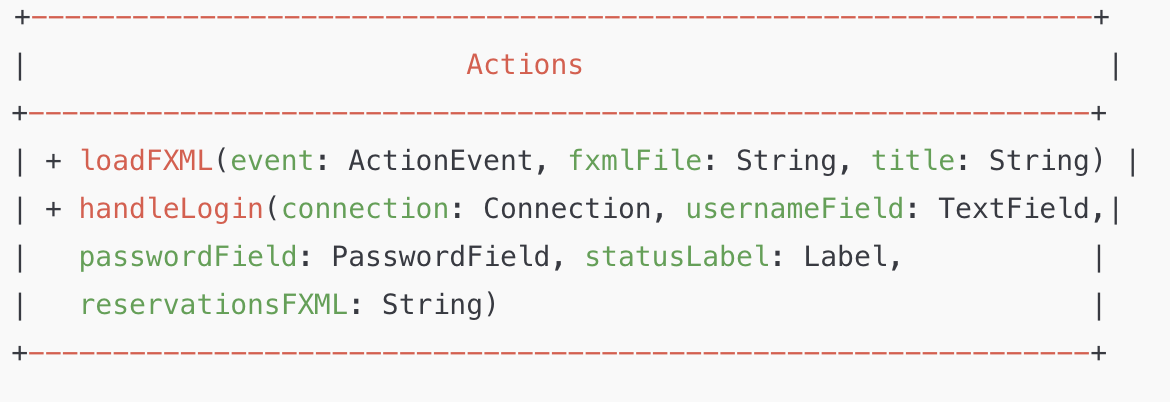
| + getAction(): String |

| + setAction(action: String): void |

| + getRole(): String |

+-------------------------------------------------------------+

12. Actions



13. RegisterController

