|  |  |
| --- | --- |
| **Project title** | **Movie Ticket Booking System** |
| **Author(s)** | **Hoza Violeta Maria** |
| **Group** | **30424** |

# Task Description

The Movie Ticket Booking System is a Java application designed for managing cinema related activities, including movie details, showtimes, user information, payments, and more.

The system accommodates two types of users, clients, and admins. Clients represent regular users who can browse movies, view showtimes, purchase tickets, and make payments. Clients have a user account with basic information such as full name, email, and phone number. Clients can view their booking history, including past and upcoming showtimes, purchased tickets, and payment details. Admins have additional privileges for managing the system. Admins can add/delete movies, update showtimes, add more locations, and access administrative functionalities. User authentication is implemented to distinguish between clients and admins.

The system supports a sign-up process where users can create an account. After successful registration, users can log in using their credentials. By registering, users gain access to personalized features such as booking tickets, viewing booking history, and leaving reviews. The system enforces validation rules to ensure accurate and valid information is provided during the sign-up process. The validation rules include:

* Required fields must be filled.
* Email addresses must follow a valid format.
* Phone numbers must be 10 digits.
* Passwords should meet specified security criteria.

Validation error messages guide users to correct any inaccuracies in their input. Upon successful validation, user data is retrieved from the form, and a **Customer** object is created. The system then attempts to insert the user into the database. The sign-up process provides immediate feedback to the user:

* If the user is successfully registered, a success message is displayed, prompting the user to log in from the main page.
* If the user already exists, a message guides them to log in from the main page.

Upon completion of the sign-up process, users are encouraged to log in using their newly created credentials. The system seamlessly integrates the sign-up and login functionalities to provide a user-friendly experience.

The login process allows registered users to access the Movie Ticket Booking System. Users are required to provide the following information in the login form:

* **Username:** The unique identifier associated with the user's account.
* **Password:** The user's password for account authentication.

The **AuthenticationService** class plays a vital role in authenticating users within the Movie Ticket Booking System. It communicates with the database to verify user credentials and determine their user type. Developers can call the **authenticateUser** method to authenticate users during the login process. The returned **UserType** indicates whether the user is an admin or a regular customer. If authentication is successful, a success message is displayed, welcoming the user. If unsuccessful, appropriate error messages are printed to the console. Upon successful authentication, the **openDashboard(UserType userType)** method is called to open the appropriate dashboard. The user is redirected to either the admin or customer dashboard based on their user type.

The **BaseController** class serves as a base class for other controllers within the Movie Ticket Booking System. It includes methods for handling common tasks related to animation and page loading. This class is designed to promote code reusability and maintainability across different controllers. Other controllers within the application extend the **BaseController** class to inherit these common functionalities, promoting a modular and organized code structure.

The AlertMaker class is a utility class for displaying alerts and confirmation dialogs. This utility class provides methods to display custom alerts and confirmation dialogs with a consistent design. It includes a method to display a simple alert with an exit button and another method to show a confirmation dialog with customizable title, header, and content. The class uses JavaFX components for creating the alert dialogs.

A screenshot of a login page

Description automatically generated

A screenshot of a computer

Description automatically generated

The class MainApp is a JavaFX Application Entry Point. This class serves as the entry point for the Movie Ticket Booking System JavaFX application. It initializes the main stage, loads the login page, and sets up the initial scene.

The User class represents a general user in the Movie Ticket Booking System. This class defines a general user with common attributes such as user ID, username, user type, and password. It includes methods to get and set these attributes. The user type is specified using the UserType enum. The Admin class represents an administrator user in the Movie Ticket Booking System. This class extends the User class and provides additional attributes specific to administrators. An administrator user only has a default constructor and a parameterized constructor, and it inherits the getters and setters for common attributes from the User class. The Customer class represents a customer user in the Movie Ticket Booking System. This class extends the User class and provides additional attributes specific to customers, such as full name, email, and phone number. It includes constructors, getters, and setters for these attributes.

The AdminHomeController serves as the main dashboard for administrators, handling navigation to different sections of the application. The CustomerHomeController serves as the main dashboard for the clients.

The MoviesAdmin class serves as a controller for the Movies Administration page, enabling users with administrative privileges to view, add, remove, and refresh the list of movies. It integrates with the DatabaseHandler class to interact with the underlying database.

The **AddMovie** class is a controller responsible for handling the addition of new movies to the Movie Ticket Booking System. It provides a user interface for inputting movie details such as title, cast, director, release date, genre, duration, rating, and additional details. Users can also attach a poster image to the movie.

The **CinemaController** class is a controller responsible for managing cinema-related functionalities within the Movie Ticket Booking System. It handles the addition, update, deletion, and display of cinemas along with their screens.

The **ShowController** class is a controller responsible for managing showtime-related functionalities within the Movie Ticket Booking System. It handles the addition, update, deletion, and display of showtimes.

The **BookShow** class is a controller responsible for managing the process of booking tickets for a show in the Movie Ticket Booking System. It handles user inputs related to seat selection, payment method, and the execution of booking and payment transactions.

The ShowMovie controller manages the display of detailed information for a selected movie. It initializes and populates the UI elements with movie details, including the name, rating, release date, cast, director, genre, and movie poster. The controller extends the BaseController for shared functionality across controllers.

The AddMovie controller manages the addition of a new movie to the database. It handles user input for movie details such as name, director, cast, release date, genre, rating, and poster. The controller ensures that all required fields are filled, validates input, and interacts with the DatabaseHandler to insert the movie into the database.

The ProfileController is a controller Class for User Profile Management. This class manages user profile information, allowing users to view and update their personal details. Users can navigate to other sections of the application such as the home page, movies page, settings, and history. It includes functionality for updating user profile information, including the username, full name, phone, and email.

The HistoryController class is a controller Class for User Reservation History Page. This class manages the functionality of the User Reservation History Page, displaying the user's past reservations. It includes a TableView to present the reservation details such as location, cinema, movie, hall, date, number of tickets, payment method, and price. The controller provides methods to update and initialize the table based on user data.

The class SettingsAdmin is a controller class for Admin Settings Page. This class manages the functionality of the Admin Settings Page, allowing the admin to update their password. It includes JavaFX elements such as buttons and password fields for interaction. The controller provides methods to handle button actions, update the admin's password, and initialize the controller.

# Class Discovery

|  |  |
| --- | --- |
| **SignUpController** | |
| Handles user sign-up. Validates user input. Interacts with the database for user registration. | Collaborates with the **DatabaseHandler** for database operations related to user registration and with the class Customer. |

|  |  |
| --- | --- |
| **LogInController** | |
| Manages the user authentication. Directs users to the appropriate dashboard. | AuthenticationService |

|  |  |
| --- | --- |
| **CinemaController** | |
| Manages addition, update, deletion, and display of cinemas along with their screens. | DatabaseHandler, Cinema, Screen |

|  |  |
| --- | --- |
| **AddMovie** | |
| Allows administrators to add movies. Validates the inputs. Interacts with the database for movie insertion. | Movie, DatabaseHandler |

|  |  |
| --- | --- |
| **ShowController** | |
| Manages the user interface for the "Shows" page. | Collaborates with FXML components for the UI (e.g., TableView, Buttons, Labels). |
| Perform actions like adding, updating, and removing shows. | Collaborates with the **DatabaseHandler** for database operations. Collaborates with the **Showtime**, **Movie**, **Cinema**, and **Screen** classes for representing and manipulating show-related data. |
| Handle user interactions, such as button clicks. | Collaborates with the **AlertMaker** for displaying alert dialogs. |

|  |  |
| --- | --- |
| **BookShow** | |
| Manages the user interface for the "Shows" page. | Collaborates with FXML components for the UI (e.g., TableView, Buttons, Labels). |
| Validate and process user inputs for booking tickets. Execute booking and payment transactions in collaboration with the database. | Collaborates with the **MoviesCustomer** class to retrieve movie, cinema, screen, and show information. Collaborates with the **DatabaseHandler**, **Reservation**, and **Payments** classes for booking and payment transactions. |
| Handle user interactions, such as button clicks. | Collaborates with the **AlertMaker** for displaying alert dialogs. |

|  |  |
| --- | --- |
| **Reservation** | |
| Represent reservations in the system. Store information about the user, the number of tickets, the associated payment, and the show. | Collaborates with the **DatabaseHandler** for database operations related to reservations. |

|  |  |
| --- | --- |
| **Movie** | |
| Represents movie details such as title, cast, director, release date, genre, duration, rating, and poster. | **DatabaseHandler,**  **MoviesAdmin, AddMovie,**  **ShowMovie** |

|  |  |
| --- | --- |
| **HistoryController** | |
| Displays the user's past reservations in a TableView. Provides methods to update and initialize the table based on user data. | DatabaseHandler, Showtime, Payment, Reservation |

# Class Diagram

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated A screenshot of a screen

Description automatically generated

A screenshot of a computer

Description automatically generated