# AirTime

EECS 3311: Group 12
Hena Patel (218109124), Ossama Benaini (), Dilpreet Bansi (218803213)

# Table of Contents



# CRC Cards

| **Class Name:** DB | |
|---|---|
| **Parent Class:** None<br>**Subclasses:** None<br>**Implements:** DBInterface | |
| **Responsibilities:**<br>➢ Database connection Establish a MySQL database connection and provide with methods for testing, setting auto-commit, and performing a close on the connection.<br>➢ **User management:** methods to create, check, retrieve, update, and delete users from the database.<br>➢ **Flight management:** Methods to add, | **Collaborators:**<br>➢ Connection from java.sql: It instigates the database connection.<br>➢ **utility:** Local utility class instance, it provides some helper functions for strings.<br>➢ **User, Flight, Booking, Transaction:** Classes that represent different entities in the booking system. They find use as return types or as parameters in |

| | |
|---|---|
| retrieve, update, and delete flight records.<br>➢ **Bookings management:** Methods for operating with bookings, including creating, getting, updating, and deleting.<br>➢ **Transaction management:** Methods to create and retrieve transactions associated with bookings.<br>➢ **Utility functions:** Checking flight availability and booking existence. | various methods. |

| |
|---|
| **Class Name:** DBInterface |
| **Parent Class:** None<br>**Subclasses:** None<br>**Implemented by:** DB |

| **Responsibilities:** | **Collaborators:** |
|---|---|
| ➢ Define the contract for CRUD operations for User, Flight, Booking, and Transaction entities.<br>➢ **User control methods:** Define methods for adding, reading, updating, and deleting user records.<br>➢ **Flight management**: Describe the methodology for managing flights, such as create, read, update, and delete.<br>➢ **Booking management:** Define methods to create, read, update, and delete bookings.<br>➢ **Transaction management:** The methodology to manage transaction create and retrieve transaction record is defined.<br>➢ **Utility functions:** Flight availability and booking existence; this supports various system operations, such as verifying if seats are available or if a user has an existing booking. | ➢ DB class: Implements this interface, providing the actual database interaction calls.<br>➢ User, Flight, Booking, Transaction classes: Represent entities that the interface manages through CRUD operations |

| **Class Name:** utility | |
|---|---|
| **Parent Class:** None<br>**Subclasses:** None | |
| **Responsibilities:**<br>➢ **Print an array's contents:** The printResult method loops through a provided array and tries to print or process elements.<br>➢ **Check for null or empty string:** The checkEmptyOrNullString method checks if any of the input string parameters are either null or empty | **Collaborators:**<br>➢ DB class: Uses utility as a helper to verify that strings like user credentials and verifies that strings are neither empty nor null before proceeding with database operations. |

| **Class Name:** Booking | |
|---|---|
| **Parent Class:** None<br>**Subclasses:** None | |
| **Responsibilities:**<br>➢ It keeps information about a booking, like booking ID, user ID, flight ID, date of booking, and the number of seats booked.<br>➢ Getters and setters provide controlled access for modification and viewing of booking details. | **Collaborators:**<br>➢ **DB class:** It will use the Booking objects to insert, read, modify and delete booking records in the database.<br>➢ **DBInterface interface:** Specifies methods that interact with booking data by using Booking either as return type or as method parameter. |

| **Class Name:** Flight |
|---|
| **Parent Class:** None<br>**Subclasses:** None |

| **Responsibilities:** | **Collaborators:** |
|---|---|
| ➢ Stores and manipulates information about a flight, including flight ID, flight number, cities of departure and destination, time of departures and arrivals, price, and number of available seats.<br>➢ The getter and setter methods allow controlled access and modification to the flight details. | ➢ **DB class:** This uses the Flight objects to create, read, update, and delete flight records within the database.<br>➢ **DBInterface interface:** Defines methods that will interact with flight data, either returning or taking in type Flight. |

| **Class Name:** Transaction | |
|---|---|
| **Parent Class:** None<br>**Subclasses:** None | |
| **Responsibilities:** | **Collaborators:** |
| ➢ Maintains information about a transaction-date, such as transaction ID, user ID, booking ID, amount of transaction, and date of transaction.<br>➢ The getters and setters provide controlled means of access and modification to the transaction information. | ➢ **DB class:** Uses Transaction objects to create, retrieve, and manage transaction records in the database.<br>➢ **DBInterface interface:** Specification of methods that interact with transaction data, using Transaction as return type or method parameter. |

| **Class Name:** User | |
|---|---|
| **Parent Class:** None<br>**Subclasses:** None | |
| **Responsibilities:** | **Collaborators:** |
| ➢ It holds the information of a user and manages their user ID, their username, password, and role.<br>➢ Getters and setters provide controlled access to and modification of user details. | ➢ **DB class:** Uses User objects to create, retrieve, update, and delete user records in the database.<br>➢ **DBInterface interface:** Defines methods related to managing a user entity, using User either as return |

| | |
|---|---|
| ➢ This assures that if the role field is null, the default role assigned would be "customer". | value or method parameter. |

| |
|---|
| **Class Name:** Main |
| **Parent Class:** None<br>**Subclasses:** None |

| | |
|---|---|
| **Responsibilities:**<br>➢ Initial Setup and Configuration of the Main Application Window<br>➢ Creates a new JFrame to host the application window, configures its layout manager and window size, and sets default close behavior.<br>➢ Manages multiple panels and provides mechanisms for navigating between different pages: WelcomePage, LoginPage, RegisterPage.<br>➢ Instantiates pages such as WelcomePage, LoginPage, and RegisterPage and adds them to the container. The users can easily move between these pages.<br>➢ Initialises and sets a Controller to handle the interaction logic between the user interface and backend services. | **Collaborators:**<br>➢ **Controller class:** Receives the CardLayout and JPanel container to manage navigation and handle user actions that change the views.<br>➢ **WelcomePage, LoginPage, RegisterPage classes:** These are the major pages users interact with in an application. The Main class is responsible for these pages inside a container.<br>➢ **JFrame, JPanel, CardLayout** |

| |
|---|
| **Class Name:** Controller |
| **Parent Class:** None<br>**Subclasses:** None |

| | |
|---|---|
| **Responsibilities:**<br>➢ Managing page navigation: It is responsible for navigating between | **Collaborators:**<br>➢ **DB:** This class provides user creation and verification, among other basic |

| | |
|---|---|
| pages, such as WelcomePage, LoginPage, and AdminDashBoardPage, through CardLayout.<br>➢ Managing which user is authenticated, creating a user through createUser, and managing current user state.<br>➢ Interacts with the DB class to fetch and store user data<br>➢ Managing user log in/log off and current session status (currentUser).<br>➢ Provides methods for navigating between different pages, including the welcome page, login page, and registration page. | database operations.<br>➢ **User class:** This represents the currently authenticated user.<br>➢ **JPanel, CardLayout classes**<br>➢ **AdminDashBoardPage, WelcomePage, LoginPage, RegisterPage classes:** UI components pages that the Controller switches between **depending on the user's action.**<br>➢ **SQLException class:** It handles database exceptions while verifying the user. |

| | |
|---|---|
| **Class Name:** WelcomePage | |
| **Parent Class:** JPanel<br>**Subclasses:** None | |
| **Responsibilities:**<br>➢ Output, a welcome message and buttons for logging in and registering.<br>➢ Page Navigation<br>➢ Event Handling | **Collaborators:**<br>➢ Controller class: Relegates the logic for navigating to the Controller, which processes page transitions based on user interactions.<br>➢ JPanel and JButton classes<br>➢ ActionListener class |

| | |
|---|---|
| **Class Name:** RegisterPage | |
| **Parent Class:** JPanel<br>**Subclasses:** None | |
| **Responsibilities:**<br>➢ It displays a registration form with User ID, Username, Role, and | **Collaborators:**<br>➢ Controller class : This will handle the register logic, calling createUser with |

| | |
|---|---|
| Password.<br>➢ Checks whether the User ID is numeric or not and provides proper error messages in case of invalid input.<br>➢ Page navigation<br>➢ This form will capture the user's input and invoke the createUser method in Controller to register the user. | form data, and navigate to WelcomePage if user wants to go back.<br>➢ JPanel, JButton, JTextField, JPasswordField classes<br>➢ ActionEvent class |

| |
|---|
| **Class Name:** LoginPage |
| **Parent Class:** JPanel<br>**Subclasses:** None |

| **Responsibilities:** | **Collaborators:** |
|---|---|
| ➢ It provides a login form with fields to enter the username and password.<br>➢ Forwards the login credentials entered to the Controller for authentication.<br>➢ This provides options for either submitting the login form or navigating back to WelcomePage if the user clicks the Back button.<br>➢ In case of an unsuccessful login attempt, it displays an error message to the user. | ➢ Controller class: This authenticates the user credentials through its login method. The Controller also does some page navigation in case of a successful or unsuccessful login.<br>➢ JPanel, JButton, JTextField, JPasswordField classes<br>➢ ActionListener class.<br>➢ JOptionPane class |

| |
|---|
| **Class Name:** AdminDashBoardPage |
| **Parent Class:** JPanel<br>**Subclasses:** None |

| **Responsibilities:** | **Collaborators:** |
|---|---|
| ➢ Buttons to perform the following functions (Log out, Access profile)<br>➢ Flight management: add flight, remove flight, update flight, browse | ➢ Controller class: Handles the logout logic<br>➢ JButton classJPanel, BorderLayout, FlowLayout, BoxLayout classes |

| | |
|---|---|
| flights<br>➢ Allows filtering by price range (minimum and maximum prices).<br>➢ Allows filtering by shortest travel time or omitting connecting flights.<br>➢ Filters by departure and arrival dates.<br>➢ It utilizes the Controller, which controls the business process logic.<br>➢ Interacts with the database for adding, removing, updating, and fetching flight information.<br>➢ Add Flight: Prompts the admin to enter information about the flight and then adds a flight using createFlight.<br>➢ Remove Flight: Asks for flight ID and removes the flight using deleteFlight.<br>➢ Update Flight: Allows to select one of the flights in the table and update its details using updateFlight.<br>➢ Browse Flights: Populates the table with flights and shows/hides it.<br>➢ Refreshes flights in the table once some changes have occurred..<br>➢ Displays error or success messages using JOptionPane.<br>➢ Handles exceptions for invalid inputs or failed operations.<br>➢ Displays a welcome message with the admin's username.<br>➢ Add and remove admin fuctionality | |

| |
|---|
| **Class Name:** CustomerDashBoardPage |
| **Parent Class:** JPanel<br>**Subclasses:** None |

| **Responsibilities:** | **Collaborators:** |
|---|---|
| ➢ Logout<br>➢ View Transactions history<br>➢ Displays flight data in a table (JTable) with a scrollable view.<br>➢ Uses the Controller to manage | ➢ Controller class: Handles the logout logic<br>➢ JButton class<br>➢ JPanel, BorderLayout, FlowLayout, BoxLayout classes |

| | |
|---|---|
| business logic and handle actions<br>➢ Fetches flight details from the database to display in the flight table.<br>➢ Refreshes the flight table when browsing flights.<br>➢ Updates layout dynamically based on user actions (show/hide table).<br>➢ Displays error messages using JOptionPane for failed operations or data loading issues.<br>➢ Displays a welcome message with the customer's username.<br>➢ Add flights to their cart and pay for their flights | |

# Software Architecture Diagram

AirTime System Architecture (MVC) Overview

The system is divided into three main layers according to MVC:

Model (M): Represnts the data and business logic. Responsible for maintaining the application state and interacting with the database.
Components:

User: User data and type of user are managed here - Admin/Customer.
Flight: Flight details such as Flight ID, Destination, Price, etc.
Booking: It stores and retrieves information on bookings.
Transaction: It stores and retrieves information on transactions made by the customer.

Interconnections: It interacts directly with the database layer for CRUD.

View (V): This is the layer representing the UI, in this case, Java Swing/JFrame
Components:

Welcome Page: Supports user log in and registration.
Register Page: A form is available on this page with which new users can register themselves.
Login Page: Existing users can verify themselves.
AdminDashboard: User interface by which the administrator can browse, add, update or remove flights. Admins can add and remove other admins

CustomerDashboard: UI used by customers to search and book flights and view booked flights. They can also add flights to their cart and view them.

Interconnections: Takes input from the Controller layer and forwards action by user to the same.

Controller (C): Serves as the intermediary between the View and the Model.

Components:

Controller: Handles admin requests and updates the Model. Handles customer interactions by invoking appropriate Model methods.

Interconnections:

Processes user inputs from the View.

Executes business logic through Model methods.

Sends responses back to the View.