

AirTime

EECS 3311: Group 12

Hena Patel (218109124), Ossama Benaini (), Dilpreet Bansi (218803213)

Table of Contents

AirTime	1
CRC Cards	2
Class Name: DB	2
Class Name: DBInterface	3
Class Name: utility	4
Class Name: Booking	4
Class Name: Flight	4
Class Name: Transaction	5
Class Name: User	5
Class Name: Main	6
Class Name: Controller	6
Class Name: WelcomePage	7
Class Name: RegisterPage	7
Class Name: LoginPage	8
Class Name: AdminDashBoardPage	9
Class Name: CustomerDashBoardPage	9
Software Architecture Diagram	10

CRC Cards

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

<p>retrieve, update, and delete flight records.</p> <ul style="list-style-type: none"> ➤ Bookings management: Methods for operating with bookings, including creating, getting, updating, and deleting. ➤ Transaction management: Methods to create and retrieve transactions associated with bookings. ➤ Utility functions: Checking flight availability and booking existence. 	<p>various methods.</p>
--	-------------------------

<p>Class Name: DBInterface</p>	
<p>Parent Class: None Subclasses: None Implemented by: DB</p>	
<p>Responsibilities:</p> <ul style="list-style-type: none"> ➤ Define the contract for CRUD operations for User, Flight, Booking, and Transaction entities. ➤ User control methods: Define methods for adding, reading, updating, and deleting user records. ➤ Flight management: Describe the methodology for managing flights, such as create, read, update, and delete. ➤ Booking management: Define methods to create, read, update, and delete bookings. ➤ Transaction management: The methodology to manage transaction create and retrieve transaction record is defined. ➤ 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. 	<p>Collaborators:</p> <ul style="list-style-type: none"> ➤ DB class: Implements this interface, providing the actual database interaction calls. ➤ User, Flight, Booking, Transaction classes: Represent entities that the interface manages through CRUD operations

Class Name: utility	
Parent Class: None Subclasses: None	
Responsibilities: <ul style="list-style-type: none"> ➤ Print an array's contents: The printResult method loops through a provided array and tries to print or process elements. ➤ Check for null or empty string: The checkEmptyOrNullString method checks if any of the input string parameters are either null or empty 	Collaborators: <ul style="list-style-type: none"> ➤ 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 Subclasses: None	
Responsibilities: <ul style="list-style-type: none"> ➤ It keeps information about a booking, like booking ID, user ID, flight ID, date of booking, and the number of seats booked. ➤ Getters and setters provide controlled access for modification and viewing of booking details. 	Collaborators: <ul style="list-style-type: none"> ➤ DB class: It will use the Booking objects to insert, read, modify and delete booking records in the database. ➤ 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 Subclasses: None	

Responsibilities: <ul style="list-style-type: none"> ➤ 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. ➤ The getter and setter methods allow controlled access and modification to the flight details. 	Collaborators: <ul style="list-style-type: none"> ➤ DB class: This uses the Flight objects to create, read, update, and delete flight records within the database. ➤ DBInterface interface: Defines methods that will interact with flight data, either returning or taking in type Flight.
--	--

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

Class Name: User	
Parent Class: None Subclasses: None	
Responsibilities: <ul style="list-style-type: none"> ➤ It holds the information of a user and manages their user ID, their username, password, and role. ➤ Getters and setters provide controlled access to and modification of user details. 	Collaborators: <ul style="list-style-type: none"> ➤ DB class: Uses User objects to create, retrieve, update, and delete user records in the database. ➤ 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 Subclasses: None	
Responsibilities: <ul style="list-style-type: none"> ➤ Initial Setup and Configuration of the Main Application Window ➤ Creates a new JFrame to host the application window, configures its layout manager and window size, and sets default close behavior. ➤ Manages multiple panels and provides mechanisms for navigating between different pages: WelcomePage, LoginPage, RegisterPage. ➤ Instantiates pages such as WelcomePage, LoginPage, and RegisterPage and adds them to the container. The users can easily move between these pages. ➤ Initialises and sets a Controller to handle the interaction logic between the user interface and backend services. 	Collaborators: <ul style="list-style-type: none"> ➤ Controller class: Receives the CardLayout and JPanel container to manage navigation and handle user actions that change the views. ➤ 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. ➤ JFrame, JPanel, CardLayout

Class Name: Controller	
Parent Class: None Subclasses: None	
Responsibilities: <ul style="list-style-type: none"> ➤ Managing page navigation: It is responsible for navigating between 	Collaborators: <ul style="list-style-type: none"> ➤ DB: This class provides user creation and verification, among other basic

<p>pages, such as WelcomePage, LoginPage, and AdminDashBoardPage, through CardLayout.</p> <ul style="list-style-type: none"> ➤ Managing which user is authenticated, creating a user through createUser, and managing current user state. ➤ Interacts with the DB class to fetch and store user data ➤ Managing user log in/log off and current session status (currentUser). ➤ Provides methods for navigating between different pages, including the welcome page, login page, and registration page. 	<p>database operations.</p> <ul style="list-style-type: none"> ➤ User class: This represents the currently authenticated user. ➤ JPanel, CardLayout classes ➤ AdminDashBoardPage, WelcomePage, LoginPage, RegisterPage classes: UI components pages that the Controller switches between depending on the user's action. ➤ SQLException class: It handles database exceptions while verifying the user.
---	--

Class Name: WelcomePage	
Parent Class: JPanel Subclasses: None	
Responsibilities: <ul style="list-style-type: none"> ➤ Output, a welcome message and buttons for logging in and registering. ➤ Page Navigation ➤ Event Handling 	Collaborators: <ul style="list-style-type: none"> ➤ Controller class: Relegates the logic for navigating to the Controller, which processes page transitions based on user interactions. ➤ JPanel and JButton classes ➤ ActionListener class

Class Name: RegisterPage	
Parent Class: JPanel Subclasses: None	
Responsibilities: <ul style="list-style-type: none"> ➤ It displays a registration form with User ID, Username, Role, and 	Collaborators: <ul style="list-style-type: none"> ➤ Controller class : This will handle the register logic, calling createUser with

Password. ➤ Checks whether the User ID is numeric or not and provides proper error messages in case of invalid input. ➤ Page navigation ➤ 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. ➤ JPanel, JButton, JTextField, JPasswordField classes ➤ ActionEvent class
---	--

Class Name: LoginPage	
Parent Class: JPanel Subclasses: None	
Responsibilities: <ul style="list-style-type: none"> ➤ It provides a login form with fields to enter the username and password. ➤ Forwards the login credentials entered to the Controller for authentication. ➤ This provides options for either submitting the login form or navigating back to WelcomePage if the user clicks the Back button. ➤ In case of an unsuccessful login attempt, it displays an error message to the user. 	Collaborators: <ul style="list-style-type: none"> ➤ 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. ➤ JPanel, JButton, JTextField, JPasswordField classes ➤ ActionListener class. ➤ JOptionPane class

Class Name: AdminDashBoardPage	
Parent Class: JPanel Subclasses: None	
Responsibilities: <ul style="list-style-type: none"> ➤ Buttons to perform the following functions (Log out, Access profile) ➤ Flight management: add flight, remove flight, update flight, browse 	Collaborators: <ul style="list-style-type: none"> ➤ Controller class: Handles the logout logic ➤ JButton classJPanel, BorderLayout, FlowLayout, BoxLayout classes

<p>flights</p> <ul style="list-style-type: none"> ➤ It utilizes the Controller, which controls the business process logic. ➤ Interacts with the database for adding, removing, updating, and fetching flight information. ➤ Add Flight: Prompts the admin to enter information about the flight and then adds a flight using createFlight. ➤ Remove Flight: Asks for flight ID and removes the flight using deleteFlight. ➤ Update Flight: Allows to select one of the flights in the table and update its details using updateFlight. ➤ Browse Flights: Populates the table with flights and shows/hides it. ➤ Refreshes flights in the table once some changes have occurred.. ➤ Displays error or success messages using JOptionPane. ➤ Handles exceptions for invalid inputs or failed operations. ➤ Displays a welcome message with the admin's username. 	
--	--

<p>Class Name: CustomerDashBoardPage</p>	
<p>Parent Class: JPanel Subclasses: None</p>	
<p>Responsibilities:</p> <ul style="list-style-type: none"> ➤ Logout ➤ View Transactions history ➤ Displays flight data in a table (JTable) with a scrollable view. ➤ Uses the Controller to manage business logic and handle actions ➤ Fetches flight details from the database to display in the flight table. ➤ Refreshes the flight table when browsing flights. ➤ Updates layout dynamically based on 	<p>Collaborators:</p> <ul style="list-style-type: none"> ➤ Controller class: Handles the logout logic ➤ JButton class ➤ JPanel, BorderLayout, FlowLayout, BoxLayout classes

user actions (show/hide table). ➤ Displays error messages using JOptionPane for failed operations or data loading issues. ➤ Displays a welcome message with the customer's username.	
--	--

Software Architecture Diagram

AirTime System Architecture (MVC) Overview

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

Model (M): Represents 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.

CustomerDashboard: UI used by customers to search and book flights and view booked flights.

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.

