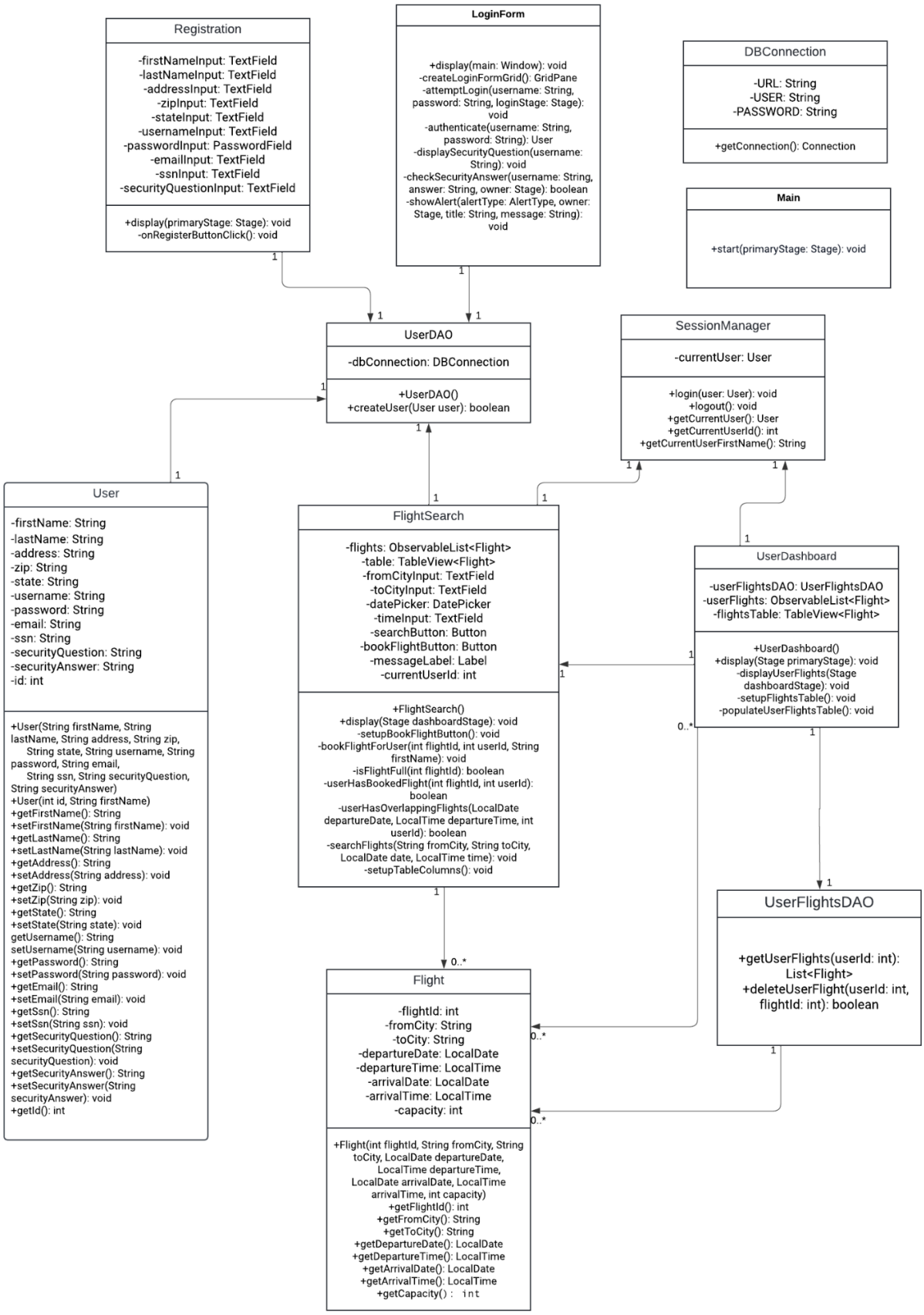
1. **Requirements and specification of the project**

The requirements for this project are to make an airline reservation system similar to Expedia or Kayak. The requirements for this project are divided into two parts: functionality and coding requirements.

For functionality, when the program is run the first thing that should be visible is a splash screen which gives the user the option to register or login. New users should always register as they are not in the database and they must provide personal information like their name, address, SSN, as well as a username and password that they can use to login. The registration page should also ask the user a security question that they can use to log in should they forget their password. After logging in the customer should have the option to search for flights, book a specific flight and delete that flight from their bookings. If they've already booked a flight then the customer should not have the option to book the same flight. The system should warn a user about flights that have conflicting dates and times and not allow the user to book conflicting flights. The application should keep track of the number of passengers on a flight and not let a user book a full flight. Users should be able to logout and login and have their booking information still there. The user should always have the option to go back to the main menu.

For coding requirements there should be at least four different class files. Out of these four concepts at least three should be utilized: encapsulation, polymorphism, inheritance, composition. At least one abstract class and one interface should be used. There should be proper comments for explanations as well as exception handling in every section of code. Database programming should be done using JDBC and the RDBMS should be one of Oracle, MySQL, PostgreSQL or SQL.

2. UML class Diagram


Registration
-firstNameInput: TextField
-lastNameInput: TextField
-addressInput: TextField
-zipInput: TextField
-stateInput: TextField
-usernameInput: TextField
-passwordInput: PasswordField
-emailInput: TextField
-ssnInput: TextField
-securityQuestionInput: TextField
+display(primaryStage: Stage): void
-onRegisterButtonClick(): void
LoginForm
+display(main: Window): void
-createLoginFormGrid(): GridPane
-attemptLogin(username: String, password: String, loginStage: Stage): void
-authenticate(username: String, password: String): User
-displaySecurityQuestion(username: String): void
-checkSecurityAnswer(username: String, answer: String, owner: Stage): boolean
-showAlert(alertType: AlertType, owner: Stage, title: String, message: String): void
DBConnection
-URL: String
-USER: String
-PASSWORD: String
+getConnection(): Connection
Main
+start(primaryStage: Stage): void
UserDAO
-dbConnection: DBConnection
+UserDAO()
+createUser(User user): boolean
SessionManager
-currentUser: User
+login(user: User): void
+logout(): void
+getCurrentUser(): User
+getCurrentUserId(): int
+getCurrentUserFirstName(): String
User
-firstName: String
-lastName: String
-address: String
-zip: String
-state: String
-username: String
-password: String
-email: String
-ssn: String
-securityQuestion: String
-securityAnswer: String
-id: int
+User(String firstName, String lastName, String address, String zip, String state, String username, String password, String email, String ssn, String securityQuestion, String securityAnswer)
+User(int id, String firstName)
+getFirstName(): String
+setFirstName(String firstName): void
+getLastName(): String
+setLastName(String lastName): void
+getAddress(): String
+setAddress(String address): void
+getZip(): String
+setZip(String zip): void
+getState(): String
+setState(String state): void
getUsername(): String
setUsername(String username): void
+getPassword(): String
+setPassword(String password): void
+getEmail(): String
+setEmail(String email): void
+getSsn(): String
+setSsn(String ssn): void
+getSecurityQuestion(): String
+setSecurityQuestion(String securityQuestion): void
+getSecurityAnswer(): String
+setSecurityAnswer(String securityAnswer): void
+getId(): int
FlightSearch
-flights: ObservableList<Flight>
-table: TableView<Flight>
-fromCityInput: TextField
-toCityInput: TextField
-datePicker: DatePicker
-timeInput: TextField
-searchButton: Button
-bookFlightButton: Button
-messageLabel: Label
-currentUserId: int
+FlightSearch()
+display(Stage dashboardStage): void
-setupBookFlightButton(): void
-bookFlightForUser(int flightId, int userId, String firstName): void
-isFlightFull(int flightId): boolean
-userHasBookedFlight(int flightId, int userId): boolean
-userHasOverlappingFlights(LocalDate departureDate, LocalTime departureTime, int userId): boolean
-searchFlights(String fromCity, String toCity, LocalDate date, LocalTime time): void
-setupTableColumns(): void
UserDashboard
-userFlightsDAO: UserFlightsDAO
-userFlights: ObservableList<Flight>
-flightsTable: TableView<Flight>
+UserDashboard()
+display(Stage primaryStage): void
-displayUserFlights(Stage dashboardStage): void
-setupFlightsTable(): void
-populateUserFlightsTable(): void
UserFlightsDAO
+getUserFlights(userId: int): List<Flight>
+deleteUserFlight(userId: int, flightId: int): boolean
Flight
-flightId: int
-fromCity: String
-toCity: String
-departureDate: LocalDate
-departureTime: LocalTime
-arrivalDate: LocalDate
-arrivalTime: LocalTime
-capacity: int
+Flight(int flightId, String fromCity, String toCity, LocalDate departureDate, LocalTime departureTime, LocalDate arrivalDate, LocalTime arrivalTime, int capacity)
+getFlightId(): int
+getFromCity(): String
+getToCity(): String
+getDepartureDate(): LocalDate
+getDepartureTime(): LocalTime
+getArrivalDate(): LocalDate
+getArrivalTime(): LocalTime
+getCapacity() : int
1
0..*

3. Data Model (Tables and all the PKs and FKs)

Database Schema: authentication_system

Users Table

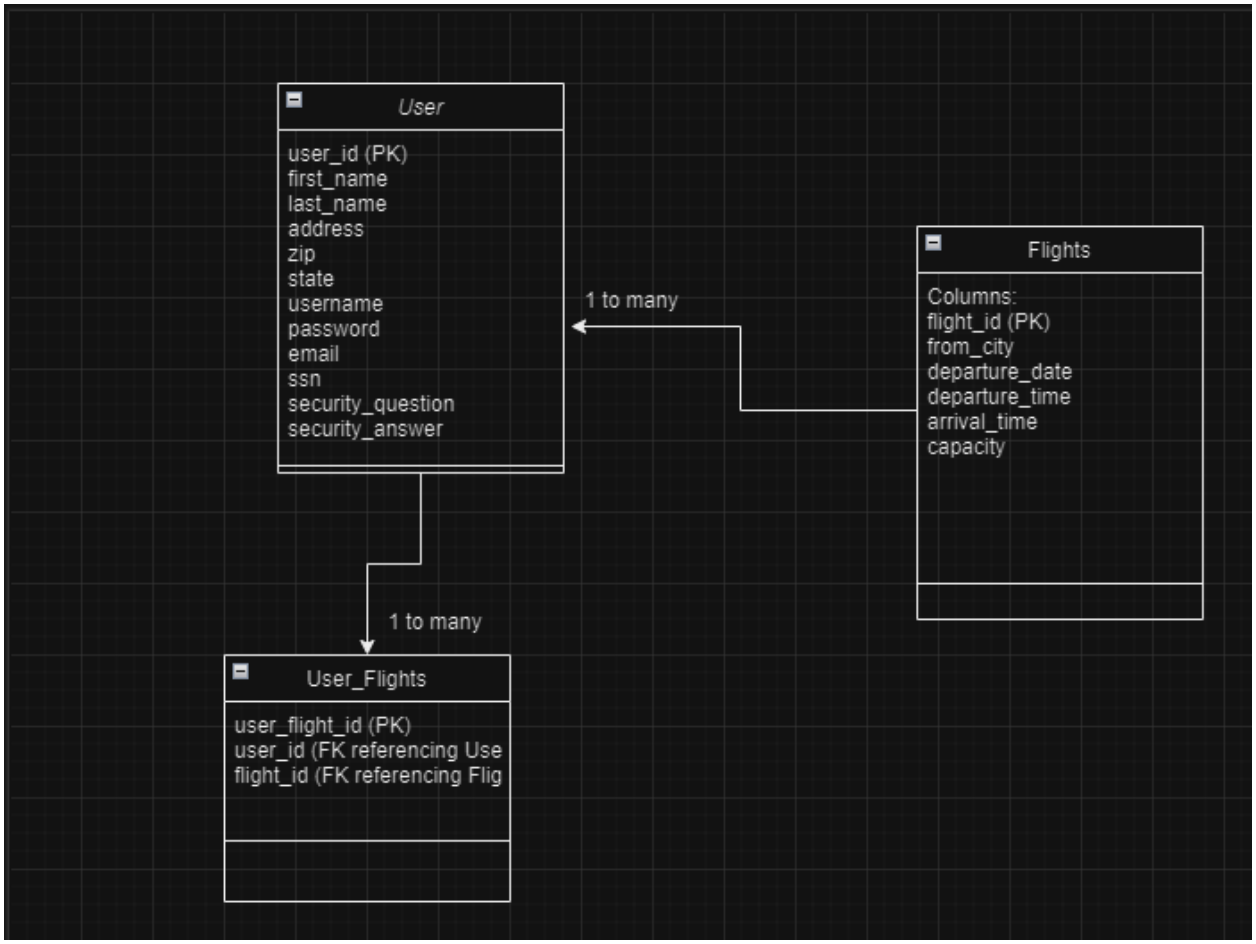
Columns:
user_id (PK)
first_name
last_name
address
zip
state
username
password
email
ssn
security_question
security_answer

Flights Table

Columns:
flight_id (PK)
from_city
departure_date
departure_time
arrival_time
capacity

User_Flights Table

Columns:
user_flight_id (PK)
user_id (FK referencing Users Table)
flight_id (FK referencing Flights Table)



4. Functionality and flow explanation

Our program is a flight reservation system designed to facilitate and manage airline tickets. It is broken up into 5 main sections.

id	first_name	last_name	address	zip	state	username	password	email	ssn	security_question	security_answer
1	a	a	a	a	a	a	a	a	a	a	a
5	asd	asd	asd	asda	sda	sd	asdasd	asda	sdas	d	d
6	tan	mai	5523 Alder Dr	302131	GA	tanmai1	Ben82055	tan@gmail.com	123456789	Alder	Alder
7	john	Doe	123 street	09321	NYC	John	Doe	j@gmail.com	2315313123	street	street
8	mac	king	5442 drive	23151	FL	mac	king	mac@gmail.com	21315123	drive	drive
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

flight_id	from_city	to_city	departure_date	departure_time	arrival_date	arrival_time	capacity
1	New York	London	2023-12-15	08:00:00	2023-12-16	08:00:00	3
2	New York	London	2023-12-15	08:00:00	2023-12-16	08:00:00	3
3	Los Angeles	Tokyo	2023-12-20	13:00:00	2023-12-21	17:00:00	3
4	San Francisco	Paris	2023-11-15	09:30:00	2023-11-16	10:00:00	3
5	Chicago	Berlin	2023-10-05	22:15:00	2023-10-06	12:30:00	3
6	Miami	Madrid	2023-09-09	07:45:00	2023-09-10	08:50:00	3
7	Dallas	Rome	2023-08-22	06:00:00	2023-08-23	07:00:00	3
8	Denver	Amsterdam	2023-07-18	20:30:00	2023-07-19	09:15:00	3
9	Boston	Dubai	2023-06-30	11:00:00	2023-07-01	19:00:00	3
10	Seattle	Beijing	2023-05-25	23:45:00	2023-05-26	23:59:00	3
11	Atlanta	Sydney	2023-04-15	15:00:00	2023-04-17	10:30:00	3
12	Houston	Sao Paulo	2023-03-10	12:00:00	2023-03-11	13:20:00	3
14	New York	London	2023-12-15	10:00:00	2023-12-15	20:00:00	3
15	New York	Paris	2023-12-15	10:00:00	2023-12-15	19:30:00	3
16	New York	Tokyo	2023-12-15	10:00:00	2023-12-16	07:00:00	3
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

user_id	flight_id	booking_date	first_name	id
1	15	2023-11-30 20:04:25	NULL	3
1	15	2023-11-30 20:25:57	tan	4
6	15	2023-12-01 18:10:04	tan	6
NULL	NULL	NULL	NULL	NULL

User Registration and Login:

- **Users need to register and create an account or log in with existing credentials.**

A database is set up to store information on registration, that is later retrieved when the user inputs login credentials.

Once logged in a new window where user can search for a flight is opened

Flight Search:

- **Users input their travel details such as departure and destination cities, dates.**

A database table with preexisting details of flights is set up that is retrieved when the user clicks search flight, a table view is then displayed with the flights.

Flight Selection:

- **The system presents a list of available flights, including details like departure and arrival times.**

The user is able to select from the list of available flights, once the user clicks book flight the information of that flight with the flight id is stored in a database for users specific flights.

Each flight has a seat capacity, and each flight can only be booked once by the same user.

Flight Deletion:

- **Users are able to view and delete booked**

If users click delete flight a new window is opened with a table view of flights that user has booked. The information retrieved is from the database of users' specific flights.

Password Recovery:

- **Users are able to recovery password using security question**

Users can retrieve password by clicking the forgot password button during the login screen. They will be prompted to enter answers to security questions. If authenticated, an alert will be displayed showcasing that specific user's password. The password is retrieved from the user information database.