**CIS 3270**

**Term Project**

**APJ**

**Airline Reservation System**

**Annie Do**

**Peihan Wang**

**John Hwang**

**12/04/2018**

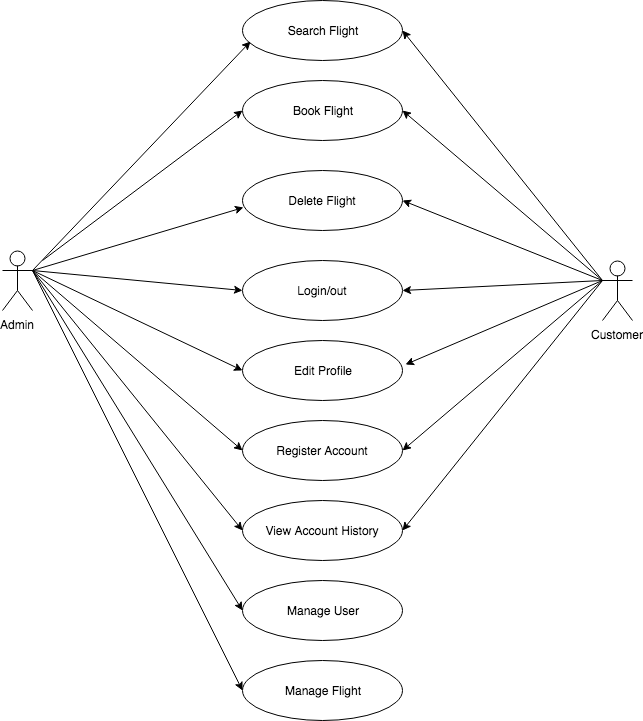
Table Content

1. Briefly explains requirements and specification of the project…………………..…….…3
2. Showing the class diagram using UML…………………………………………………...4
3. Showing the Data Model (Tables and all the PKs and FKs)...............................................5
4. Briefly explains about functionality and flow of the program…………………………...10
5. **The requirements and specification of the project**

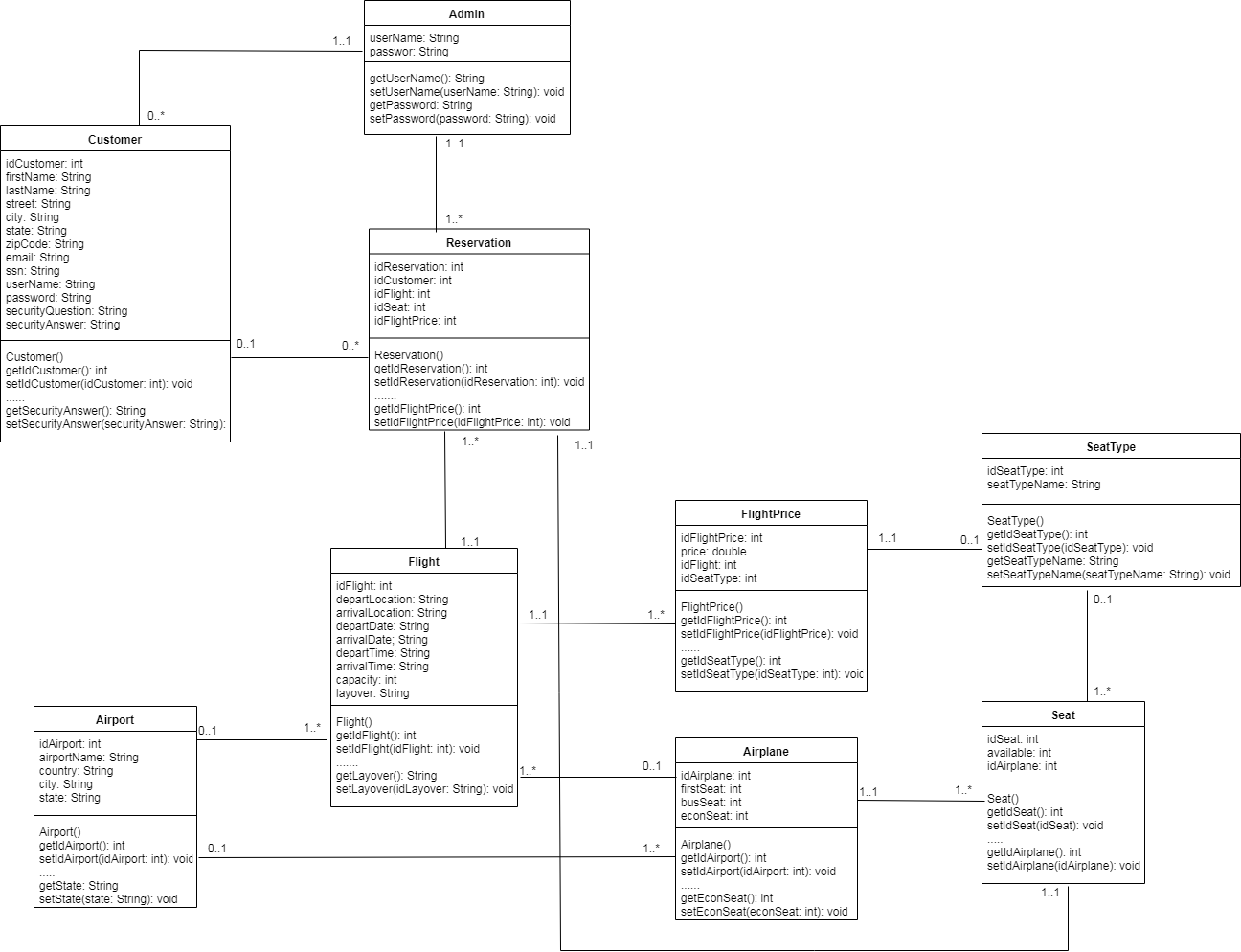
This project is making an airline reservation system application. We name our application is APJ. The application will include:

* The application start with flash screen
* It will link to Log In page
* From Log In page, if user is Admin, it will link to Admin management page. If user is Customer, it will link to Customer booking page.
* If User does not have account, they need to click Register to sign up. In Register Page will require customer to fill out first name, last name, address, zip, state, username, password, email, SSN and a security question for password recovery.
* If User forgets password, they will be able to retrieve his password knowing the username with the security question.
* Admin has more privileges regarding this application.
* For Customer, After login into application a customer should be able to:
* Search flight database for flights based on from city/ to city/ date and time of flight.
* Be able to book a flight and add that to his account.
* Be able to delete a flight from his account.
* Customer can not book the same flight more than once.
* If there is a conflict about date and time of a flight, application will warn the Customer about this and not able to book the conflicting flight.
* Application should keep track of number of passengers booked at each flight and should not let a user book the flight if it is full.
* Admin will be able to do all customer activities in addition to be able to add, update or delete a flight.
* User can logout and login; previously booked reservation should be there in his/her account.
* User always be able to go back to Main Menu.

Overall, the application should be functional in most parts considering the requirements mentioned above.

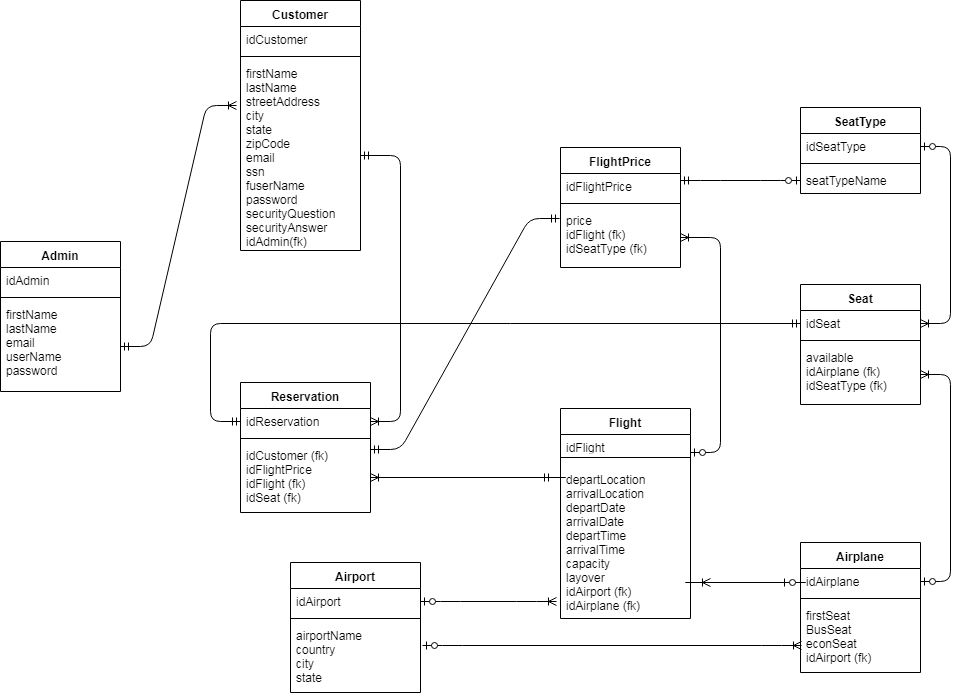


**II. The class diagram using UML**

****

**III. The Data Model**

* ERD

****

* **Table FK and PK**

Table name: Admin

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute Name | Type | Key | Nullable | Referential Constraint |
| idAdmin | numeric(4) | PK (AI) | No |  |
| firstName | varchar(50) |  | No |  |
| lastName | varchar(50) |  | No |  |
| userName | varchar(50) |  | No |  |
| password | varchar(50) |  | No |  |

Table Name: Customer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute Name | Type | Key | Nullable | Referential Constraint |
| idCustomer | numeric(4) | PK (AI) | No |  |
| firstName | varchar(50) |  | No |  |
| lastName | varchar(50) |  | No |  |
| streetAddress | varchar(100) |  | Yes |  |
| city | varchar(50) |  | Yes |  |
| state | varchar(50) |  | Yes |  |
| zipCode | varchar(50) |  | Yes |  |
| email | varchar(50) |  | No |  |
| ssn | varchar(50) |  | No |  |
| userName | varchar(50) |  | No |  |
| password | varchar(50) |  | No |  |
| securityQuestion | varchar(100) |  | No |  |
| securityAnswer | varchar(100) |  | No |  |
| idAdmin | numeric(4) | FK referencing Admin.idAdmin | No | On DELETE Cascade  ON UPDATE Cascade |

Table name: Airport

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute Name | Type | Key | Nullable | Referential Constraint |
| idAirport | numeric(4) | PK (AI) | No |  |
| airportName | varchar(50) |  | No |  |
| country | varchar(50) |  | No |  |
| city | varchar(50) |  | No |  |
| state | varchar(50) |  | Yes |  |

Table name: Airplane

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute Name | Type | Key | Nullable | Referential Constraint |
| idAirplane | numeric(4) | PK (AI) | No |  |
| firstSeat | numeric(5) |  | No |  |
| busSeat | numeric(5) |  | No |  |
| econSeat | numeric(5) |  | No |  |
| idAirport | numeric(5) | FK referencing Airport.idAirport | No | On DELETE Cascade  ON UPDATE Cascade |

Table name: Flight

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute Name | Type | Key | Nullable | Referential Constraint |
| idFlight | numeric(4) | PK (AI) | No |  |
| departLocation | varchar(100) |  | No |  |
| arrivalLocation | varchar(100) |  | No |  |
| departDate | varchar(50) |  | No |  |
| arrivalDate | varchar(50) |  | No |  |
| departTime | varchar(50) |  | No |  |
| arrivalTime | varchar(50) |  | No |  |
| capacity | numeric(5) |  | No |  |
| layover | varchar(50) |  | Yes |  |
| idAirport | numeric(4) | FK referencing Airport.idAirport | Yes | On DELETE Cascade  ON UPDATE Cascade |
| idAirplane | numeric(4) | FK referencing Airplane.idAirplane | Yes |  |

Table name: SeatType

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute Name | Type | Key | Nullable | Referential Constraint |
| idSeatType | numeric(4) | PK (AI) | No |  |
| seatTypeName | varchar(50) |  | No |  |

Table name: FlightPrice

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute Name | Type | Key | Nullable | Referential Constraint |
| idFlightPrice | numeric(4) | PK (AI) | No |  |
| price | double() |  | No |  |
| idFlight | numeric(4) | FK referencing Flight.idFlight | No | On DELETE Cascade  ON UPDATE Cascade |
| idSeatType | numeric(4) | FK referencing SeatType.idSeatType | No | On DELETE Cascade  ON UPDATE Cascade |

Table name: Seat

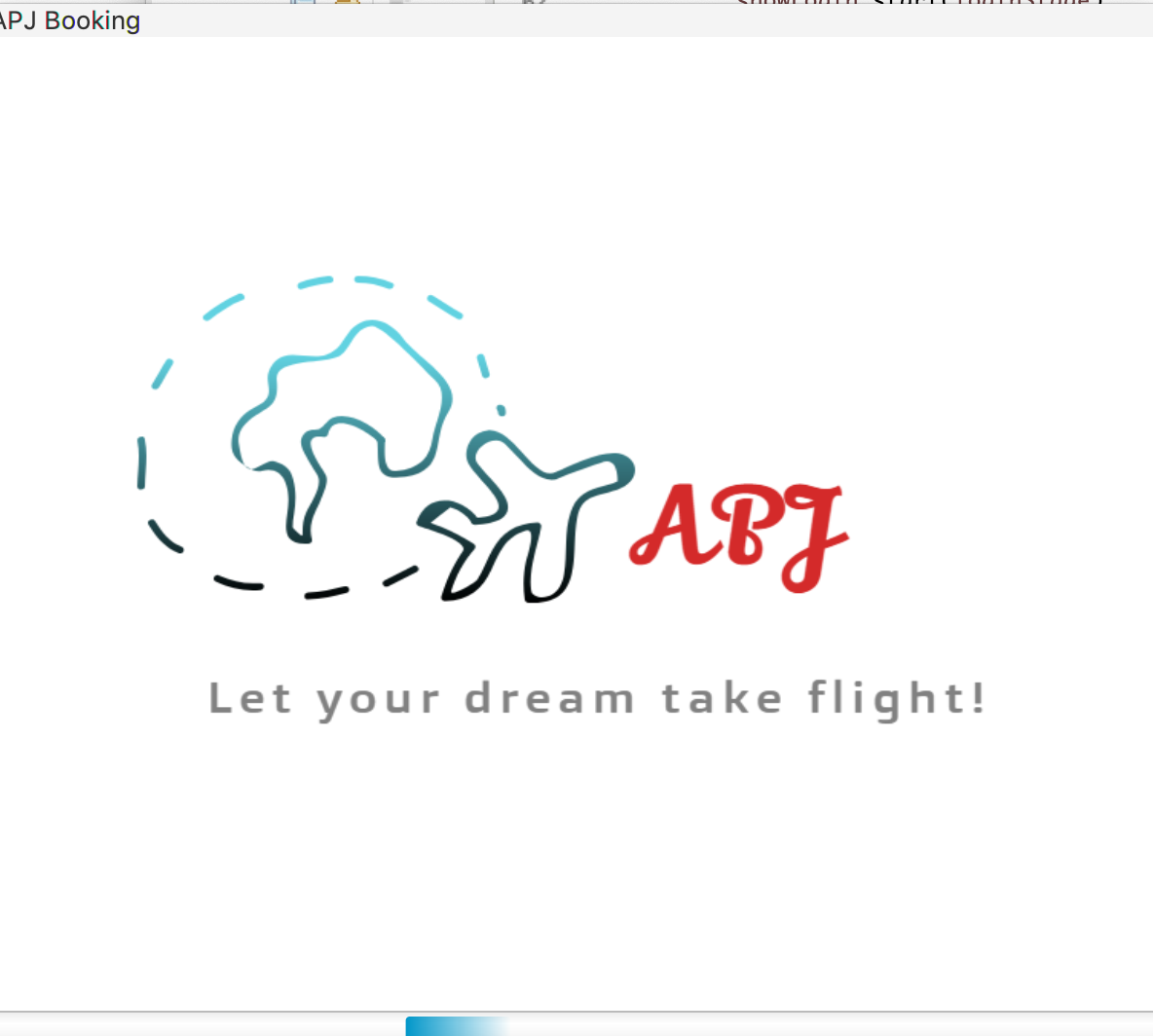
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute Name | Type | Key | Nullable | Referential Constraint |
| idSeat | numeric(4) | PK (AI) | No |  |
| available | numeric(5) |  | No |  |
| idAirplane | numeric(4) | FK referencing Airplane.idAirplane | No | On DELETE Cascade  ON UPDATE Cascade |
| idSeatType | numeric(4) | FK referencing SeatType.idSeatType | No | On DELETE Cascade  ON UPDATE Cascade |

Table name: Reservation

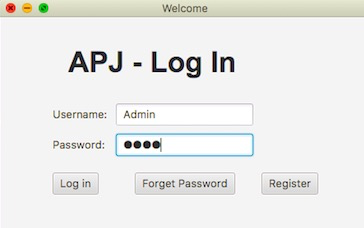
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute Name | Type | Key | Nullable | Referential Constraint |
| idReservation | numeric(4) | PK (AI) | No |  |
| idCustomer | numeric(4) | FK referencing Customer.idCustomer | No | On DELETE Cascade  ON UPDATE Cascade |
| idFlightPrice | numeric(4) | FK referencing FlightPrice.idFlightPrice | No | On DELETE Cascade  ON UPDATE Cascade |
| idFlight | numeric(4) | FK referencing Flight.idFlight | No | On DELETE Cascade  ON UPDATE Cascade |
| idSeat | numeric(4) | FK referencing Seat.idSeat | No | On DELETE Cascade  ON UPDATE Cascade |

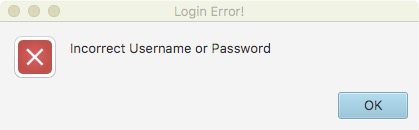
**IV. Explains about functionality and flow of the program**

1. The APJ application start with flash screen, and then it will move to Log In screen for user to log in or sign up

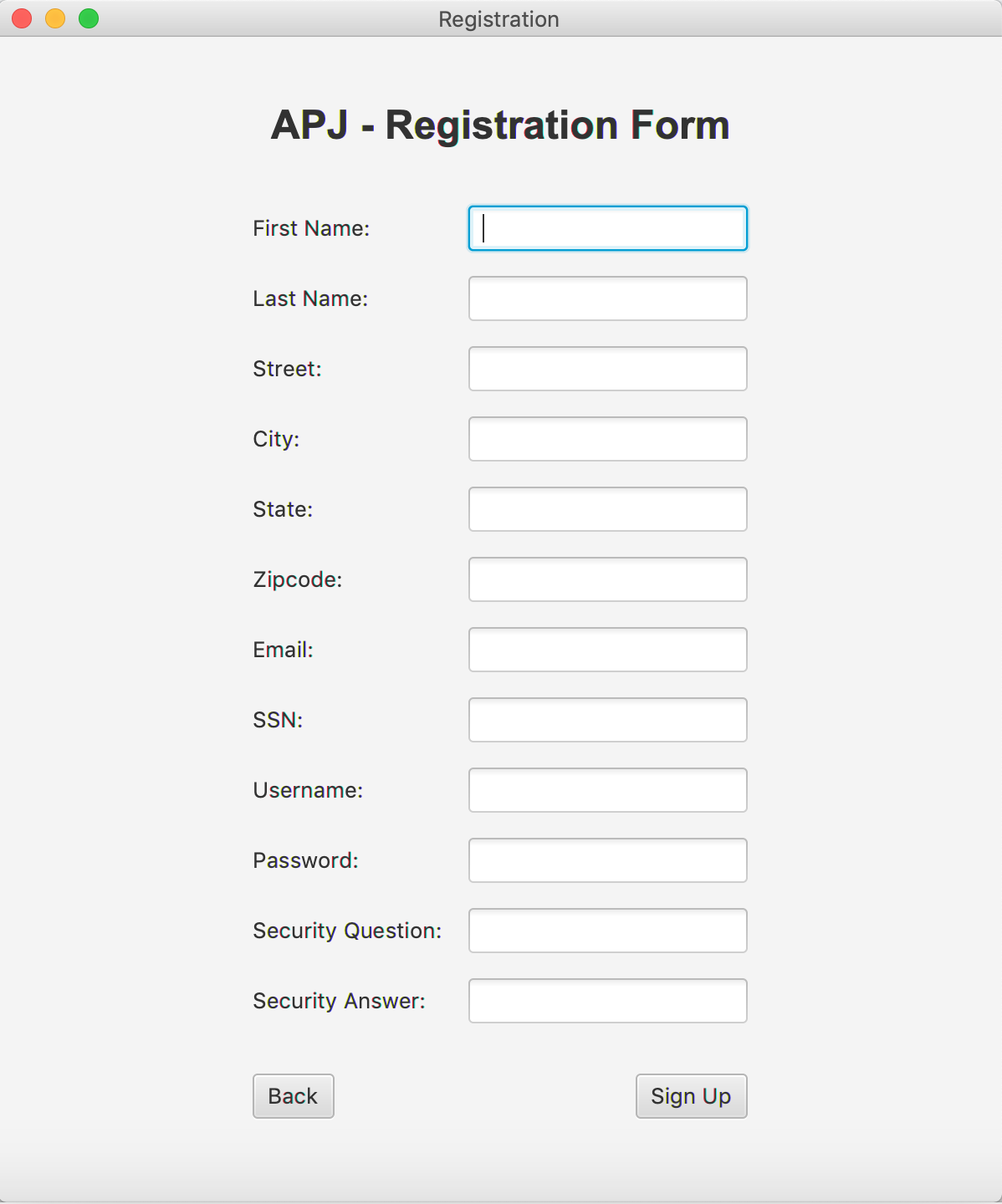


1. Log In will require a username and password that will be validated against a database.

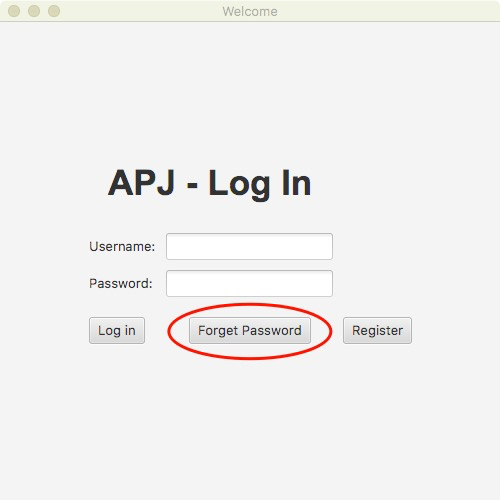




1. Users can be distinguished between customer and admin. Users will be registered as a customer by default.
2. Registration accepts the following information and will be stored into the database



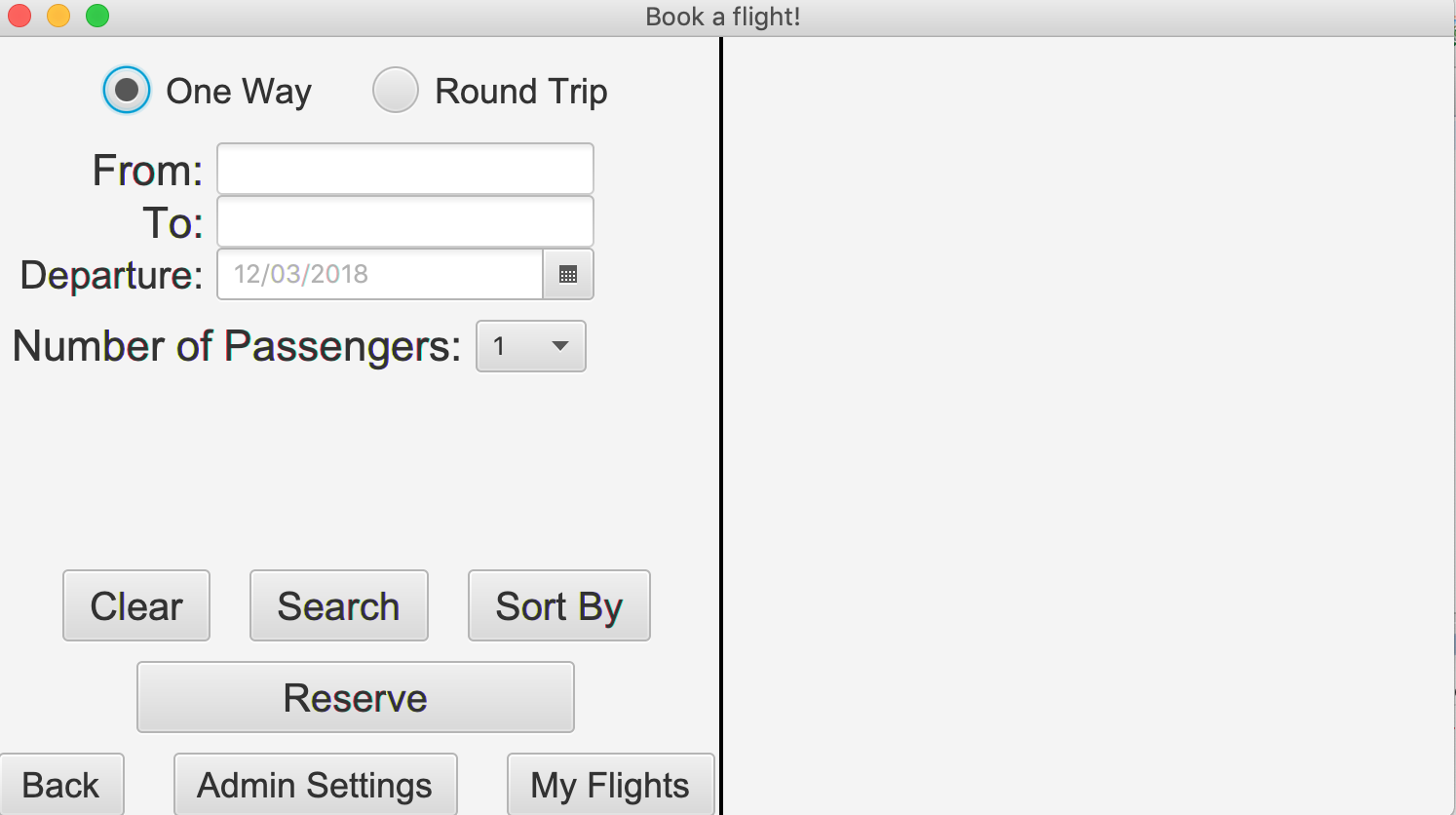
1. Recovering password will require existing username that will be validated by the database.(Click the “forgot password” button).

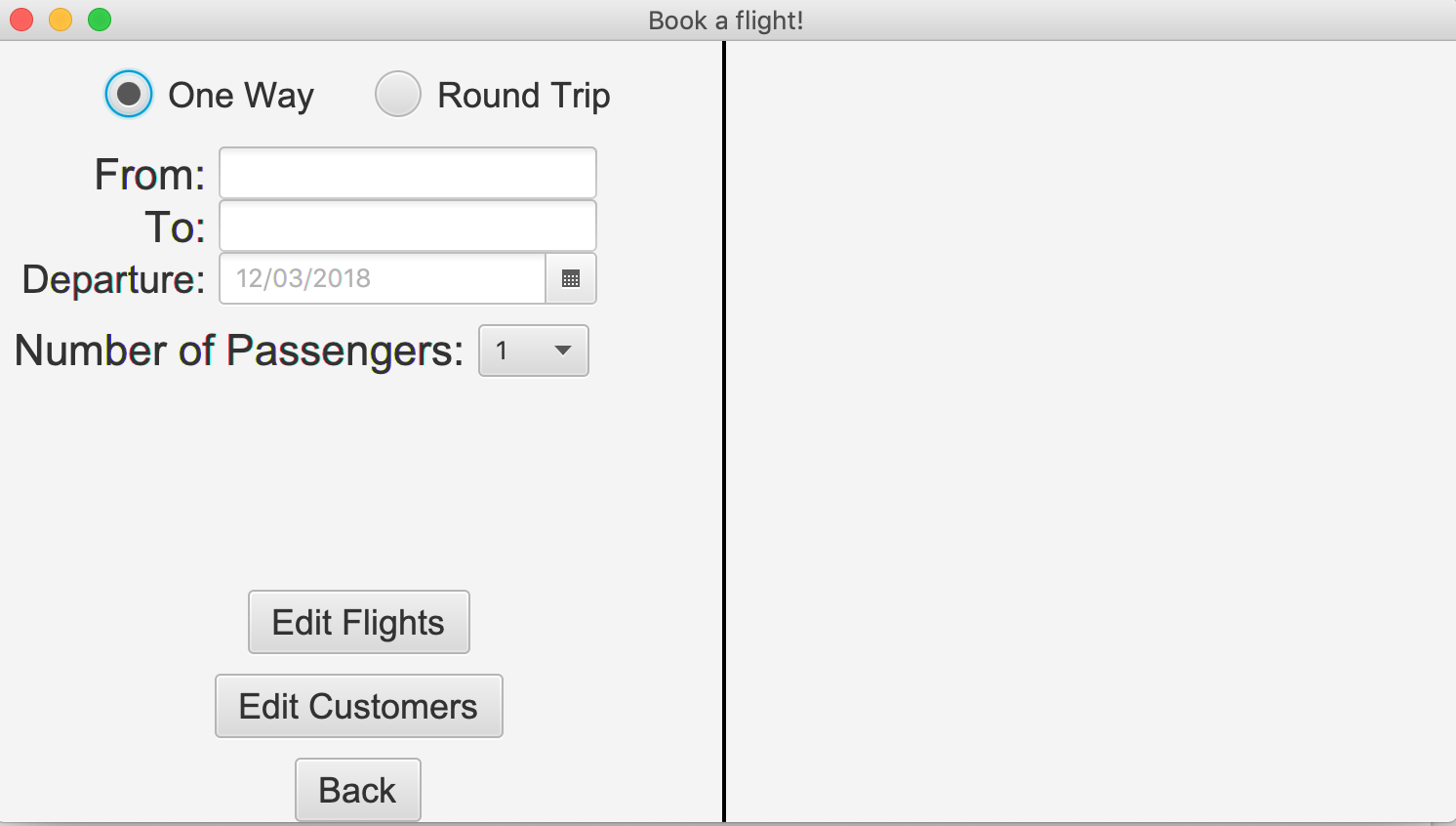


1. If username is validated by database, it will display the security question for the user and a field to enter the answer to the security question.
2. If the security question answer is correct, an alert box displaying the correct password for the account will be displayed.

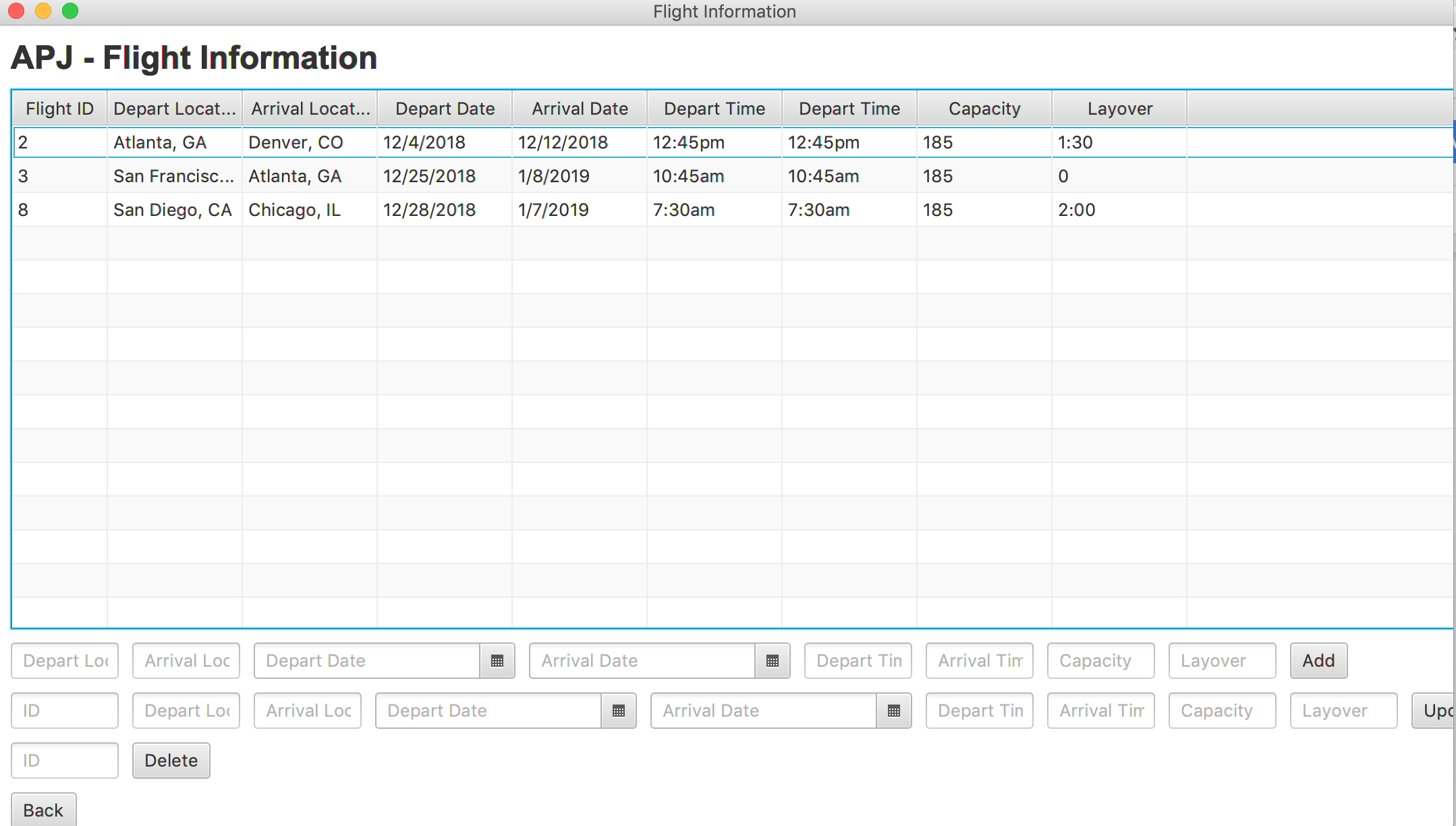


1. While logged in to the booking page all flights booked by a user will be displayed on the main page
2. User can enter flight number from their list of booked flights and delete them from their account from their main page.
3. User can refresh their booked flights.
4. Users can click on a search flights button from the booking page to take them to a search page.
5. From the search page users can enter search term into a text field and sort by search categories. results will be displayed in listview.
6. Users can book a flight by entering flight number they would like to book into the add flight textbox and clicking the add flight button.
7. If a user has already booked the selected flight or the flight time conflicts with already booked flight, user will be notified accordingly via alert box.
8. If the user is an admin, their main page will have the option to go to an add fight page, or an edit/delete page.

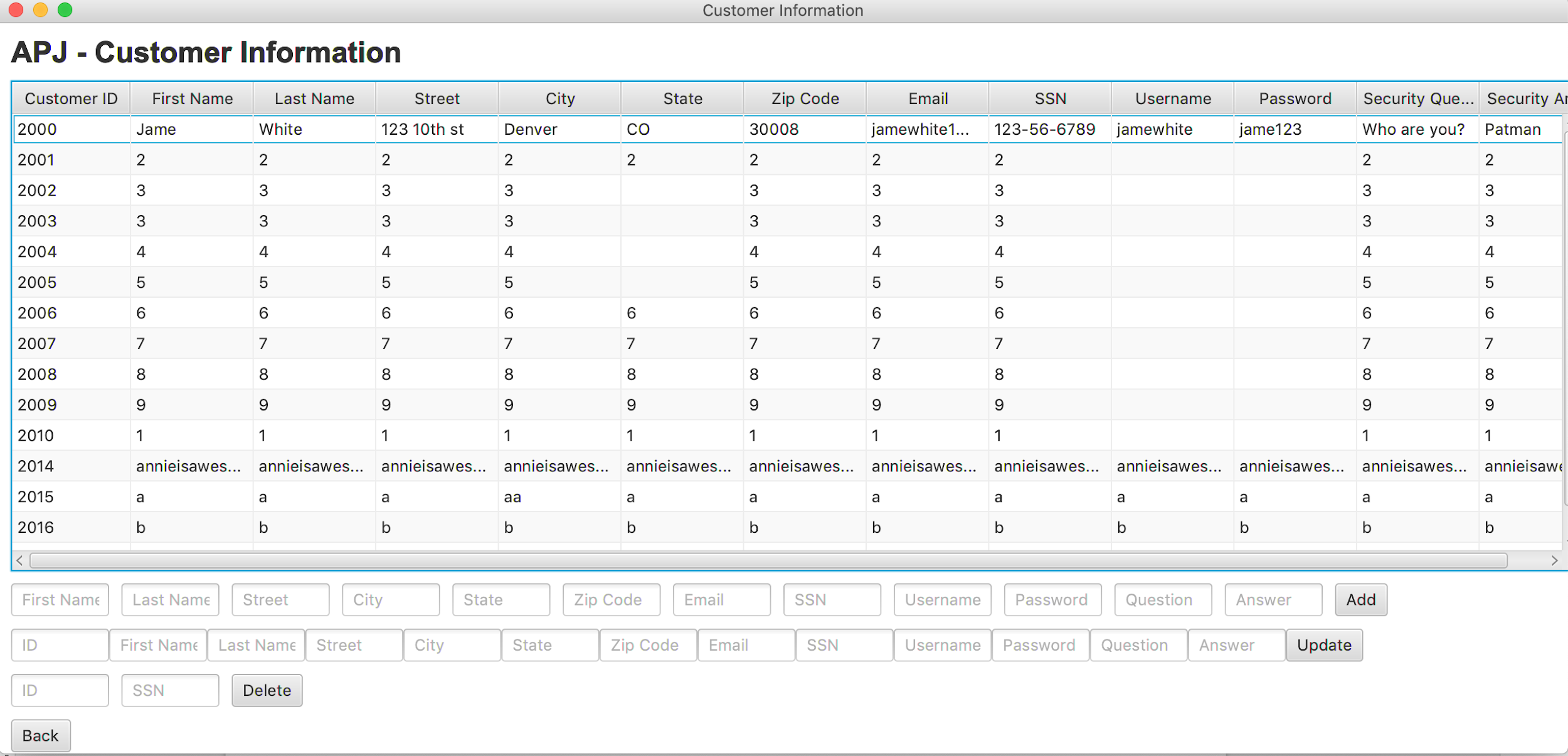




1. Click the edit flight button, it will link to flight details, Admin able to add, update and delete flight.
2. If the departure date and time is set to after the arrival date and time, it will warn the user of a conflict and will not let them create the flight.



1. Click the edit customer button, it will link to customer details, Admin able to add, update and delete customer.



1. If username already exist, it will pop up error message that Admin need to add different username for customer.
2. All pages will have an option to back to the login page.
3. The user can log out to the login page from booking page