1. The first route handler, hello(), simply returns the rendered template index.html when the user navigates to the root URL of the application.
2. The logout() route handler clears the user session and redirects them to the root URL.
3. The userlogin() route handler simply returns the rendered template user-login.html when the user navigates to the URL /User/user-login.html.
4. The userregister() route handler simply returns the rendered template user-register.html when the user navigates to the URL /User/user-register.html.
5. The booking() route handler returns the rendered template user-flight-booking.html when the user navigates to the URL /User/user-flight-booking.html.
6. The search\_flights() routing handler handles the request sent by the user when submitting the flight search form. It retrieves the form data from the request, constructs an SQL query based on the search criteria, and executes the query using a cursor. function checks if each parameter exists, and if it does, it means that the user entered this condition, and then adds this condition to the query statement. Finally the cursor will return the resulting flights as a list to be displayed in the index.html template.
7. The userloginAuth() route handler handles the POST request sent when a user submits a login form. It takes the form data from the request, encrypts the password, performs a query to retrieve the user information from the database, and creates a session for the user if the login credentials are valid. The username, email and "type" are stored in the session. If the credentials are invalid, an error message is displayed on the user-login.html template.
8. The function 'userregisterAuth' authenticates the registration details entered by the user. If the request method is POST, it grabs information from the registration form, including email, name, password, address, passport information, and date of birth. The function then hashes the password and checks if the user with the entered email address already exists in the database. If the user does not exist, the function inserts the registration details into the 'Customer' table and commits the transaction. If the user entered phone numbers, the function inserts them into the 'Customer\_Phone\_Number' table. Finally, the function renders the 'index.html' template.
9. Routing for the user's main page ("/user-main"), which requires authentication (user must be logged in). It first retrieves the user's username and email from the session. Then, it creates a cursor to execute a query to retrieve all the flights the user has booked that have not yet departed (i.e. the departure date and time is in the future). The query joins the "Ticket" and "Flight" tables to get the flight details, and uses a subquery to filter the flights based on the user's email address and future departure and arrival times. The results are stored in the "flights" variable and passed to the "User/user-main.html" template along with the username.
10. /registerAuth: This route handles user registration. It first gets user input data from the HTML form and hashes the password using SHA-256. Then, it checks if the user already exists in the Customer table. If not, it inserts the user information into the table along with the phone number(s) in the Customer\_Phone\_Number table.
11. /user-main: This route is used to render the user's main page after login. It first retrieves the user's email from the Flask session, then queries the Flight and Ticket tables to get all future flights that the user has booked. The query only selects flights that have a departure time and arrival time greater than the current time aka future flights. The resulting data is passed to the HTML template for rendering.
12. /purchase-form/<flight\_number>: This route handles the purchase form for a specific flight. It takes the flight\_number as a parameter and renders the user-payment.html template.
13. /purchase\_result: This route renders the user-purchase-result.html template.
14. /CSS/<path:path>: This route serves CSS files. It returns a 404 error if the requested file is not a CSS file.
15. /user-track: This route is used to render the user's tracked flights. It first retrieves the user's email from the Flask session, then queries the Flight and Ticket tables to get all past flights that the user has booked. The query only selects flights that have an arrival time less than the current time. The resulting data is passed to the user-track.html template for rendering.
16. /rate-result: This route renders the user-rate-result.html template.
17. /user-spend: This route is used to display the user's spending for the past year and past 6 months. It first retrieves the user's email from the Flask session, then queries the Ticket and Purchase tables to get the total spending for the past year and the monthly spending for the past 6 months. The monthly spending is calculated using the calendar module to determine the start and end dates of each month. The resulting data is passed to the user-spend.html template for rendering.
18. /user-search-flights: Use Case: A user wants to search for flights based on various criteria like flight number, departure/arrival time, and airport code. SQL: The code builds a dynamic query based on the search criteria provided by the user. After it gets input from the frontend page, it will select flight by Not None conditions. It selects the required flight information from the Flight and Ticket tables, joins them and filters the results based on the search criteria.
19. /user-book-flight: Use Case: A user wants to search for available flights based on various criteria and book a flight using their payment information. SQL: The code builds a dynamic query based on the search criteria provided by the user. It selects the required flight information from the Flight table and filters the results based on the search criteria. Once the user selects a flight, the code inserts the payment and ticket information into the Payment\_Information, Ticket and Purchase tables.
20. /user-rate-and-comments: Use Case: A user wants to rate and comment on a flight they have taken. SQL: The code first selects the flights that the user has taken from the Ticket and Flight tables. Once the user selects the flight they want to rate and comment on, the code checks if the user has already rated and commented on the flight. If they have not, the code inserts the rating and comment into the Evaluation table.
21. /track-user-spend-specifically: Use case: The system allows users to query their spending within a specified time range and view the breakdown of their spending by month. Users enter a start date and an end date, and the system retrieves all purchase records within that time frame. It then calculates the total amount spent during that period and the amount spent per month. SQL: The system uses a query to find the total value of all the user's purchase records within the specified time frame. It does this by joining the Ticket and Purchase tables and using the SUM function to calculate the total amount spent. Then, for each month within the specified time range, the system calculates the total amount spent during that month using a similar query. The results are returned as a dictionary containing each month and its corresponding consumption totals.
22. stafflogin(): This function handles the request for the staff login web page. It simply renders the staff-login.html template and returns it to the client.
23. staffregister(): This function handles the request for the staff registration page. It simply renders the staff-register.html template and returns it to the client.
24. staffLoginAuth(): This function handles the request for staff login authentication. It first retrieves the username and password from the form data submitted by the user. It then hashes the password using SHA-256 encryption. It then creates a database cursor and executes a SELECT query to check if the username and hashed password match a record in the Airline\_Staff table. If a match is found, the function creates a session for the user and redirects them to the staff main page. If a match is not found, the function returns an error message to the login page.
25. staffRegisterAuth(): This function handles the request for staff registration authentication. It first retrieves the form data submitted by the user, including the username, password, first name, last name, email, phone number, date of birth, and airline name. It then hashes the password using SHA-256 encryption. The function checks if the username already exists in the Airline\_Staff table. If it does, the function returns an error message to the registration page. If the username does not exist, the function checks if the airline name exists in the Airline table. If it does not, the function returns an error message to the registration page. If both the username and airline name exist in the database, the function inserts a new record into the Airline\_Staff table with the hashed password and other user information. It also inserts the user's email addresses and phone numbers into the Staff\_Email and Staff\_Phone\_Number tables, respectively.
26. staff\_main(): Use case: This function is responsible for displaying a list of flights that are scheduled in the next 30 days and belong to the airline the staff member is associated with. SQL: The function retrieves the airline name from the Airline\_Staff table using the staff member's username, then selects all flights that match that airline name and have a departure date within the next 30 days. The results are returned to the staff-main.html template for rendering.
27. view\_flights(): Use case: This function renders the view-flights.html template, which displays a form for searching for flights. SQL: This function does not interact with the database.
28. search\_airline\_flights(): Use case: This function is responsible for executing a search query based on the search criteria specified in the view-flights.html form and displaying the results. SQL: The function constructs a SQL query based on the form input, selecting flights from the Flight table that match the specified criteria. The results are returned to the view-flights.html template for rendering.
29. view\_customers(): Use case: This function retrieves a list of customers who have purchased tickets for a particular flight and displays it to the staff member. SQL: The function selects all customers who have purchased a ticket for the specified flight from the Customer and Ticket tables. The results are returned to the view-customers.html template for rendering.
30. create\_new\_flight(): Use case: This function renders the new-flight.html template, which displays a form for creating a new flight. SQL: This function does not interact with the database.
31. insertNewFlight(): Use case: This function is responsible for inserting a new flight into the Flight and Fly tables. SQL: The function first verifies that the flight does not already exist, that the airports and airplanes specified in the form exist in the Airport and Airplane tables respectively, and then inserts a new row into the Flight table with the specified details. It also inserts a row into the Fly table with the flight's airline name, flight number, departure date and time, airplane identification number, and flight status.
32. success(): This function renders a success page after a successful operation. Use Case: This function is used to display a success message after a successful operation such as adding a new flight, updating flight status, or adding a new airplane.
33. change-status(): This function renders a page for staff members to change the status of a flight. Use Case: This function is used by staff members to change the status of a flight, such as from "On Time" to "Delayed" or "Canceled".
34. changgleStatus(): This function changes the status of a flight in the database. Use Case: This function is used by staff members to change the status of a flight in the database. SQL: This function executes an SQL query to update the flight\_status field of a row in the Fly table, based on the user's input.
35. add-airplane(): This function renders a page for staff members to add a new airplane. Use Case: This function is used by staff members to add a new airplane to the database..
36. insertAirplane(): This function adds a new airplane to the database. Use Case: This function is used by staff members to add a new airplane to the database. SQL: This function executes an SQL query to insert a new row into the Airplane table, based on the user's input. The input is airline\_name, identification\_number, number\_of\_seats, manufacturing\_company, manufacturing.
37. displayAirplane(): This function displays all airplanes belonging to a specific airline. Use Case: This function is used by staff members to view all airplanes belonging to their airline. SQL: This function executes an SQL query to retrieve all rows from the Airplane table where airline\_name matches the session variable for the staff member's airline.
38. addAirport() and insertAirport(): This pair of functions allow staff members to add a new airport to the database. The addAirport() function renders an HTML template containing a form for entering airport information. Upon submitting the form, the insertAirport() function is called, which inserts the airport information into the Airport table in the database if it does not already exist. If the airport already exists, an error message is displayed. The SQL statements in insertAirport() use the INSERT INTO command to add a new row to the Airport table.
39. viewRatings() and displayRatings(): These functions allow staff members to view the ratings and comments submitted by customers for a particular flight. The viewRatings() function renders an HTML template containing a form for entering flight information. Upon submitting the form, the displayRatings() function is called, which retrieves the ratings and comments for the specified flight from the Evaluation table in the database. The function also calculates the average rating for the flight. SQL: The SQL statements in displayRatings() use the SELECT command to retrieve data from the Evaluation table based on the input flight information. The function also uses the AVG() function to calculate the average rating.
40. view\_report(): This function allows staff members to view a report on the airline's performance, including the number of tickets sold, revenue, and most frequent customer. The view\_report() function retrieves this information from the Ticket and Purchase tables in the database and calculates the relevant metrics. SQL: The SQL statements in view\_report() use the SELECT command to retrieve data from the Ticket and Purchase tables. The function uses the SUM() function to calculate the revenue and the COUNT() function to calculate the number of tickets sold.
41. ticket\_report(): This function allows staff members to generate a report on the airline's ticket sales between two specified dates. The ticket\_report() function retrieves the total amount of ticket sales between the specified dates from the Ticket and Purchase tables in the database. SQL: The SQL statements in ticket\_report() use the SELECT command to retrieve data from the Ticket and Purchase tables based on the input dates.
42. view\_customer\_flight\_page() and viewCustomerFlight(): This pair of functions allow staff members to view all flights taken by a particular customer on their airline. The view\_customer\_flight\_page() function renders an HTML template containing a form for entering customer email addresses. Upon submitting the form, the viewCustomerFlight() function is called, which retrieves the flight information from the Flight and Ticket tables in the database. SQL: The SQL statements in viewCustomerFlight() use the SELECT command to retrieve data from the Flight and Ticket tables based on the input customer email address and airline name.