

Team-30(target-1)

Team Members:

Erik Herbert-Hernandez dbs040

Kang Miao dbs072

Alejandro Moreno dbs078

Jinangkumar Shah dbs103

Goal of our website:

The main goal for this project was to create a user-friendly website, highlighting the use of SQL, Javascript, and HTML to bring the backend and frontend development into one project to showcase the kind of ingenuity that can be forecasted in website development. Our website is meant to display aspects practicality and aesthetics to make the user feel as comfortable when booking their flights during the already stressful time of travel planning.

How to populate tables:

-Use tables.sql

-Open hw3

-Open database using "psql -d COSC3380"

-Run \i tables.sql

Note:tables.sql is already uploaded as part of submission

****We have only populated flights_info, airport_info, flight_details and passenger tables. Initially, there are no bookings. Bookings are expected to be done by the user. Follow the instruction about page functionality below after you are successfully able to run the code.**

How to run:

- Create a folder, name it anything you like. Add these files into the folder.
- Then, create a folder named "public".
- After that, it should like the figure below:

 public	11/28/2021 3:51 AM	File folder	
 app	11/29/2021 10:26 AM	JavaScript File	23 KB
 package	11/13/2021 6:57 PM	JSON File	1 KB
 package-lock	11/13/2021 6:57 PM	JSON File	45 KB

=>To install Dependencies:





1. Open terminal right where you have package.json
2. Run command npm install to install all dependencies.
3. "dependencies": {

```

    "cors": "^2.8.5",
    "express": "^4.17.1",
    "pg": "^8.7.1"
  }

```

- Now inside the public folder, add the following files.
- Then, create a folder named “css”.
- After that, it should like the figure below:

	css	11/24/2021 2:49 PM	File folder	
	about	11/29/2021 10:26 AM	Chrome HTML Do...	1 KB
	booking	11/29/2021 10:26 AM	Chrome HTML Do...	7 KB
	checkIn	11/29/2021 10:26 AM	Chrome HTML Do...	2 KB
	create	11/29/2021 10:26 AM	Chrome HTML Do...	3 KB
	index	11/29/2021 3:55 PM	Chrome HTML Do...	2 KB
	main	11/29/2021 10:26 AM	JavaScript File	13 KB

- Inside the css folder, put “style.css” inside:

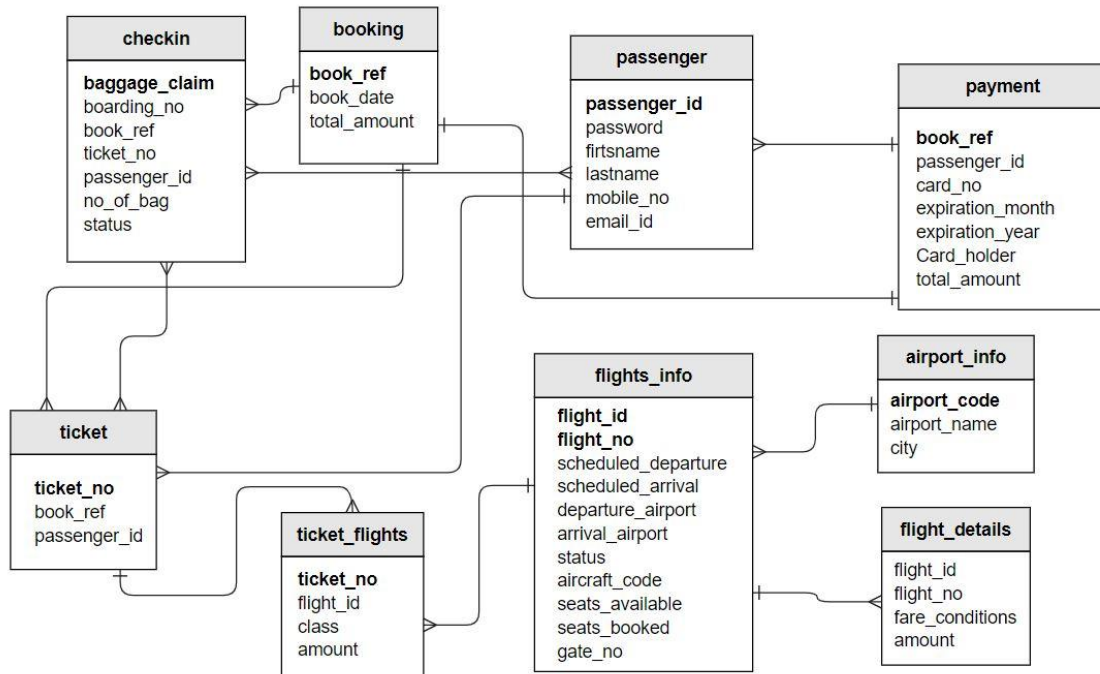
	style	11/29/2021 10:26 AM	CSS File	2 KB
---	-------	---------------------	----------	------

All set to run the code.

- Again, open terminal and run command “node app.js”
- Open browser, write <http://localhost:5000/> in the search bar

ER diagram

Airline Database ER



=>Primary keys are displayed as **bold text**.

=>Relation between tables are shown using arrows.

One to one relation =>

One to many relation(other way around for many to one) =>

Many to many relation =>

Details about Normalization of tables

=>In flight_details table flight_id,flight_no,class work as composite key.

=>Table flights_info violates 3NF because status has transitive dependency with other columns because currently all flights status are set as "scheduled".

=>Table flight_details violates 2NF because flight_id and flight_no are partially dependent on amount while fare_conditions had no functional dependency with amount. (note: flight_id,flight_no,fare_conditions work as composite PK, and if we remove fare_conditions from composite key then there will be duplicate PKs)

=>checkin table violates 3NF because we are generating 2 different baggage claims if a passenger had 2 bags. While other data associated with passengers is the same which creates transitive dependency.

=>passenger, ticket, ticket_flights, airport_info, booking are in BCNF.

How to use search page:

1. Enter arrival city such as "houston"
2. Enter departure city such as "New york"
3. Choose date range
4. Finally choose flight type direct or indirect and tap search

***Our flights are scheduled for 12/10/2021 so choose a date range that includes 12/10.**

***A good example to search for direct flights is Houston to New York.**

***A good example to search for indirect flights is Houston to Los Angeles.**

****After searching flight, YOU NEED TO REMEMBER FLIGHT ID**

=>Backend

-Search page uses primary flights_info table, joining with airport_info to compare the city names (entered by user) with city code and flight_details for fares per class.

How to use New user page:

***Only create a new ID if you are not already in the system.**

***Users are expected to remember their passenger ID once they create a new ID. later you can use it for booking.**

***Each person needs to have their own passenger ID, such as if you are trying to book a ticket for a family of 4 people then you need to have a unique passenger ID for each person in the family.**

1. Enter First name.
2. Enter Last name.
3. Enter password.(Length is restricted to 30)
4. Enter email.
5. Enter Mobile number.
6. Once you tap the Create button, you get an alert, showing you your passenger ID which you need to remember for further bookings.

=>Backend

-New user inserts into passenger table and returns auto incremented passenger ID.

=>There are initially 10 users created in the database. You can create a new user to check the functionality of the page.

How to use Booking page:

*All passengers must have a passenger ID prior to book flights.

*You must remember type of flight(direct/indirect)

*You must remember the ID of the flight that you are going to book.

1. Select flight type(direct/indirect)
2. Enter flight ID1 if flight type is direct, Enter flight ID1 & flight ID2 if type is indirect
3. Select number of passenger(up to 6)
4. Enter passenger ID for each passenger.
*Passenger ID1 will be associated with payment so use the passenger ID1 that you want to associate with payment.
5. Select class.
6. Enter credit card number
7. Select expiry month and year
8. Enter security code. (Currently our payment portal only works with security code with length of 3 digits so if your card has length of 4, I am afraid it will not work with our payment portal.Sorry for inconvenience.
9. Finally, enter the name of the card holder.
- 10.Once you tap the book button, we will display booking reference(it will be the same for the entire transaction), ticket number and passenger ID. If you are booking for an indirect flight then you will have 2 ticket numbers, first ticket number associated with your passenger ID is for first flight and second ticket number is for second flight. Although the book reference stays the same throughout the transaction.

=>backend

->First we get the amount of fare from the flight_details table joining it with the flights_info table and then we multiply it with the number of passengers. If the flight type is indirect then we search for both flight's fare using the same method then we add them together and to get the total amount we multiply it with the number of people. Now, we have all the information to insert into the booking table, book_ref is a sequence so we insert all the values and get book_ref number in return.

->Now, we insert into ticket table, we already have book_ref number, we have passenger_id from user, and ticket_no is a sequence that generates a unique 8 digit number. So using this information we insert into the ticket table to get the ticket number in return to insert data into the ticket_flights table. For ticket_flights table we already have generated ticket number, we have flight_id from user input, we have class from user input and we have already calculated fare amount per person, so we

simply insert that value in ticket_flights table. This process gets done for every passenger using a loop. If flighttype is indirect then there will be two ticket numbers for each passenger and two different flight IDs so we have added 2 more inserts into the statement that insert necessary information in ticket and ticket_flights table.

->Next process is to insert payment information to the payment table. We have used book_ref as foreign key and primary key which will define all payment information uniquely. We have associated payment methods with passenger ID1. We get the credit card information from user input. We have already counted total_amount so we simply add all the data to the payment table.

->The last process is to update seats in the flights_info table. We simply add the number of people to sets_booked and we subtract the number of people from seats available. Again, if flight type is indirect then this process is done separately for each flight ID.

=>Initially there were no bookings. You can book flights using IDs that are already in the database(from 1 to 10), or you can create new IDs, and use it for booking.

How to use check in page:

1. Enter booking reference.
2. Enter passenger ID
3. Enter ticket no.
4. Enter Number of baggage(up to 2)
5. Once you tap the check-in button, you will get the boarding number and baggage claim number for your each bags.

*Every person needs to check in separately

*Every passenger is expected to check in only once per ticket no.

*If your flight has connection then there will be 2 ticket no. so you need to check in for both ticket no. separately.

=>Backend

->To update check in information to checkin table, we get book_ref number, passenger ID, ticket number from user and we ask number of bags. We insert those information to table and generate unique baggage claim number for each bags. We generate boarding no. and output it on the checkin page which works as a boarding pass.

Also we display gate no. while the user checks in. If the flight type is indirect then the user needs to check in again for a second flight.

How to use Cancellation page:

=>You can not cancel a specific ticket. If you are cancelling, we will cancel the entire transaction.

1. Enter your booking reference number.
2. Tap cancel button.

3. Page will display a success message if you have entered the correct booking reference.
4. Page will display an unsuccessful message if the book reference is not found.

=>backend

-We first get the ticket number from the ticket table associated with booking reference, then we get flight ids associated with booking number from ticket_flights table. We determine direct/indirect flight by the count of distinct flights from the ticket_flights table and determine no of passengers.

-Then we first delete ticket_no from ticket_flights.

-Finally, we delete book_ref from checkin,payment,booking,ticket table where book_ref is equal to user input.

Error Handling(From Database):

=>Currently we are only printing **errors** from **DBMS** to **terminal** so If a user enters any values such as if the user enters a passenger ID for booking that does not exist in our database then it will violate referential integrity and database will produce error because passenger ID is FK for the tables that manage bookings.

Contact us:

For any question regarding running code or functionality of the page please contact any team member via teams or email.