AST6 Airline Online Service

**Team 6**

Kunda Wu

Thong Le

John McGinley

**Instructor**

Dr. Mike Wu

## Project Overview

Our objective is to develop a web-based application for stakeholders who have recently founded an airline named Airline Services Team-6 (or AST6) and are in need of a website that can provide a booking service to their customers. Because they are new in the market, they want their website to be released as fast as possible, and they are giving us 3 months to accomplish their expectations. Needless to say, they consider efficiency, security, reliability and scalability of the utmost importance. A small list of the features that they find absolutely essential:

* Client:

1. UI is responsive, simple but modern.
2. Console interface for users to manage their flight, special offers and flight history
3. Search functionality that allows users to sort their options by price/layovers/etc

* Server:

1. Fast and accurate response and handling logic operation based on user’s input.
2. Connect client-server workflow to database management system

* Database:

1. Scalable: not specific
2. Secure: user’s information
3. Reliable: data should be available all time
4. Consists of a collection of data related to flights (costs, number of seats, etc)

## 

## System Environment

Client

(ReactJS)

HTTP request

query

Home server

return data

HTTP response

Business layer

(express.js)

Data access layer (mysql

-connector)

Database

(MySQL)

Server (2-layer)

### Figure 1: 3-tier Architecture diagram

* Hardware & Software:

1. Apache2
2. Visual Studio Code (latest)
3. OpenVPN

* RDBMS: MySQL community server (v8.0.17)
* Host: Home server
* Frameworks & its core dependencies

1. Node.js (v11.10.1), npm (v6.7.0)
2. Front-end: React (v3.0.1), Redux (latest)
3. Back-end: express.js (v4.16.1) and mysql (connector, v2.17.1)

## Functional Requirements

1. **Users and how users access your system**

There is only one type of user which is customer/ client to our airline services. All the flight information and processes will be pre-set and simulated.

\*\* Users must sign up to start using our services.

* 1. Sign-Up

Users will be prompted to enter some information:

* + - First name, middle initial and last name
    - Gender
    - Date of Birth (DOB)
    - Email. This is important because users will have to use email to sign in our services, to get email confirmation (see 2-factor authentication, page 5), or to reset their password
    - Password (8 characters, 1 special character)
  1. Sign-In
  + Users can sign in using registered email and password
  + If users set 2-factor authentication, they will have to enter a confirmation code sent to their registered email
  1. Reset Password
* The function invokes when users click the ‘Forgot password’ button.
* It will then ask the user to provide the email to receive reset-password link.

1. **Functionality/features, functional processes and I/O(s)**

User Dashboard: When users sign in, they will have access to dashboard where they can search for flights, reserve flights, change their profile and as well as access to flight management

* + - 1. Search for flights:
* Users can pick departure and destination, pick depart time and arrival time
* Users can refine search options such as special offers applied, day/ night time, small/ big plane, extra services
* Users can prioritize flight fares - sort the given list of potential flights by the following criteria:

i. Based on price range (Allow users to set min and max price)

ii. Based on price sort (lowest to highest or vice versa)

ii. Based on duration/number of stops

iii. Based on flight classes (First Class, Economy Plus, or Economy)

* + - 1. Reserve flight: After users complete flight search and ready to pick up flight line
* Users can start to reserve it by entering more personal information such as passport/ID and credit card information. Then, users will be prompted to confirm or cancel the reservation

I. Confirm: Users will receive reservation receipt via registered email

II. Cancel: Reservation will be cancelled

* Finally, users will be redirected back to state of searching for a flight
  + - 1. Profile edit:

Users can update their password, set their preferences:

* + - 2-factor authentication: more secure for users’ account, using email confirmation whenever they sign in
    - Preferred Flight Class: next time they search, their flight class will be set as their default, they can change it anytime
    - Services: preferred services users want to be provided along their flights
    - Alert: alert users about the status of their next flight
      1. In flight management: users can view information about their flights booked with our airline.
* The information can be time, status (delay, ontime, postpone).
* Users can use this feature to cancel (Cannot be done before 24 hours prior to departure)
* Users can add more services to their flight (upgrade their class)
  + - 1. Emergency check in:
* Users will be directed to page displayed with the most early next flights
* Users can pick and fill up passport and credit card information to get in that flight

## Non-functional Issues

* **Graphical User Interface (GUI)**

We will be building a Graphical User Interface for this application using React, Redux, HTML and CSS3/Bootstrap. To use our application to search for flights, users will have to create an account with sign up form using an ‘Enroll now’ button that will be placed in the center of the page. If they are not signed in and try to access via direct url, they will be redirected to the Welcome page where users can sign in using the form with their registered email and password. Also in the Welcome page, users can reset their password with ‘Forgot password’ using their registered email address. This will allow us to send them a verification email and allow them to change their password.

Once users sign in, they can search for flights using our search bar which will be placed at the top of the page. When they search, they will fill in their departing location and desired destination, as well as departure and return dates. Users will then click search or press enter to start querying for their flights, then the page will display a list of flights matching the given criteria in our database. Users can open up advanced search to refine some search options as well as prioritization for time in day and flight classes. Moreover, users will be able to sort the returned flights by total duration/number of stops and price, they will also be able to set the min and max price when searching so they do not have to view tickets outside of their budget. Having these search options will allow users to find a flight that they need in the most streamlined way. In addition, users can change their profile by clicking a button in the upper right part of user dashboard. This will bring users to profile page that allows them to change their password and change their preferences. There will be a menu on the left of the page that allows them to do things such as manage their flight information, change or upgrade their flight, check-in their flight, or cancel entirely.

* **Security and miscellaneous**
  1. Authorization:

The security of users will be protected by email address and password. Enrolled user​ ​account​ ​information​ ​will​ ​be​ ​securely​ ​stored​ ​in​ the database.​ ​​​In​ ​order​ ​to​ ​login​ ​to the​ ​system​, the ​user​ ​must​ ​provide​ ​an​ ​existing​ ​email​ ​account​ ​along​ ​with​ ​the associated​ ​password.​ Meanwhile, if a user happens to forget password, the user have to reset his password before signing in to the system. ​The​ database system ​will​ ​be​ ​implemented​ ​using​ ​local host ​providing​ connection​ ​to​ ​server.

* 1. Secure sign-in: two-factor authentication (a confirmation code will be sent to user’s registered email)
  2. User’s information is encrypted (simple encryption) before persisting into the database system
  3. Simulated flight information

Each flight will have a maximum number of seats and each person can not double-book the same flight. When the user enters a search query, a series of desired flights will be displayed, with the number of seats left. If the number of seats is zero, that flight will not be shown to the user, even if it meets their other criteria.

* 1. Maintainability and scalability

System will be designed so it will be easy to maintain and scale up when the business grows

Also, it depends on further requirements from the stakeholders

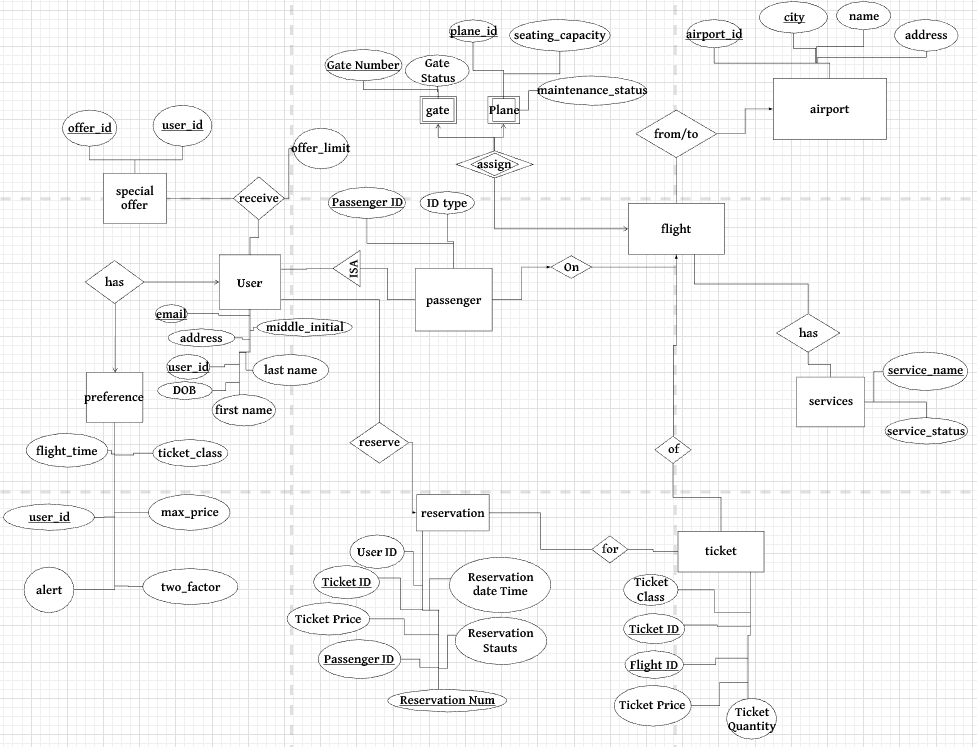
* 1. Reliability of booking/flight information

All the reservation/ flight information will be persisted and secured

* 1. Performance of this booking service (Times, overall experience, and etc)

In this AST6 project, we implement the concurrently popular technology Node.js and React to build the client side and server side.

## **E/R Diagram**

Figure 2: E/R diagram for AST6 Airline Online Services

## **Relation Schemas with description**

1. **user(user\_id [Member #], email, joined\_datetime, first\_name, last\_name, middle\_initial, DOB, address):** Each user will have to provide email, first name, last name, middle name initial, Date of Birth (DOB), and address in order to sign up, and has a unique user ID which is assigned by System

Example Tuples:

(M2206, [john@gmail.com](mailto:john@gmail.com), 10/6/2019, John, McGinley, P, 2/12/1997, 123 Fake Street)

(M2207, kunda[@gmail.com](mailto:john@gmail.com), 11/6/2014, Tyler, Daas, D, 4/3/1995, 1203 20th Street)

(M2208, tee[@gmail.com](mailto:john@gmail.com), 1/5/2009, Thong, Le, Q, 5/15/1992, 1203 Austin Street)

(M2209, jake[@gmail.com](mailto:john@gmail.com), 2/5/2015, Kunda, Wu, P, 4/12/1990, 1203 Kim Street)

(M2210, blake[@gmail.com](mailto:john@gmail.com), 10/6/2014, John, Wu,  R, 8/30/2005, 1203 Rut Street)

(M2211, rock[@gmail.com](mailto:john@gmail.com), 4/3/2011, Rock, Lee, N, 2/2/1950, 13 7th Street)

(M2212, kyle[@gmail.com](mailto:john@gmail.com), 5/9/2013, Kyle, Hertsch, A, 3/5/1960, 123 8th Street)

(M2213, andre[@gmail.com](mailto:john@gmail.com), 3/3/2015, Andre, Giant, R 8/5/1963, 143 first Street)

(M2214, blake[@gmail.com](mailto:john@gmail.com), 6/6/2006, Blake, Bortles, C, 9/4/1975, 101 West Adams St)

(M2215, doug[@gmail.com](mailto:john@gmail.com), 10/6/2019, Doug, Adams, L, 9/16/1996, 927 Hamilton St)

(M2216, juan[@gmail.com](mailto:john@gmail.com), 10/6/2019, Juan, Lopez, D, 1/1/2000, 940 Black St)

(M2217, tim[@gmail.com](mailto:john@gmail.com), 10/6/2019, Tim, Allen, Q, 5/5/2005, 409 Market St)

(M2218, tyler[@gmail.com](mailto:john@gmail.com), 10/6/2019, Tyler, Adams, C, 1/4/1959, 440 Telegraph Ave)

(M2219, maria[@gmail.com](mailto:john@gmail.com), 10/6/2019, Maria, Ramirez, D, 11/1/1991, 904 Test Dr)

(M2220, ramirez[@gmail.com](mailto:john@gmail.com), 10/6/2019, Kim, Ramirez, S, 1/5/1990, 753 Edde St)

1. **preference(user\_id, flight\_time, ticket\_class, alert, two\_factor, max\_price):** Each registered user has exactly one list of preferences, so it is a one-one relationship. A list of preferences helps user to pre-select their options in flight search (flight time (day/night), ticket class (Business class/ Economic class). Alert is Yes/No to notify user when flight status changes. Two factor is Yes/No to provide more secure to user’s account. Max price is to set a limit for a ticket price

(M0001, dayandnight, economic, yes, yes, 200)

(M0002, dayandnight, economic, yes, yes, 200)

(M0003, dayandnight, economic, yes, yes, 200)

(M0004, day, economic, yes, yes, 200)

(M0005, night, economic, no, yes, 200)

(M0006, night, business, yes, yes, any)

(M0007, night, business, yes, yes, any)

(M0008, day, business, yes, yes, any)

(M0009, night, economic, yes, no, 150)

(M0010, night, economic, yes, no, 150)

(M0011, night, economic, yes, yes, any)

(M0012, night, economic, yes, yes, 150)

(M0013, night, economic, yes, yes, 150)

(M0014, dayandnight, economic, yes, no, 150)

(M0015, dayandnight, economic, yes,no, 150)

**3. user\_offer(user\_id, offer\_id, limit\_count)**: Each offer for user has limit-of-use defined in special\_offer relation

(012206, 1010, 5)

(012207, 1011, 1)

(012208, 2010, 3)

(012209, 1310, 7)

(012210, 1052, 100)

(012211, 1092, 2)

(012212, 3921, 50)

(012213, 5901, 10)

(012214, 9012, 30)

(012215, 0019, 21)

(012216, 3910, 12)

(012217, 8491, 15)

(012218, 5019, 1)

(012219, 3828, 3)

(012220, 5231, 5)

4. **passenger(passenger\_id, ID [driver's license|Passport number]**, **ID\_type,** **first\_name, last\_name, middle\_initial, DOB)**: passenger whose information is used to reserve a flight’s ticket, and passenger can be an unregistered user. ID must be unique and  encrypted

(1, DL000000, DL, 10/6/2019, John, McGinley, P, 2/12/1997)

(2, PP122222, PP, Tyler, Daas, D, 4/3/1995)

(3, DL126666, DL, Thong, Le, Q, 5/15/1992)

(4, DL122222, DL, Kunda, Wu, P, 4/12/1990)

(5, DL125555, DL, John, Wu,  R, 8/30/2005)

(6, DL124444, DL, Rock, Lee, N, 2/2/1950)

(7, PP123456, PP, Kyle, Hertsch, A, 3/5/1960)

(8, PP133333, PP, Andre, Giant, R 8/5/1963)

(9, PP144444, PP, Blake, Bortles, C, 9/4/1975)

(10, PP222222, PP, Doug, Adams, L, 9/16/1996)

(11, DL123456, DL, Juan, Lopez, D, 1/1/2000)

(12, DL123456, DL, Tim, Allen, Q, 5/5/2005)

(13, PP123457, PP, Tyler, Adams, C, 1/4/1959)

(14, DL123456, DL, Maria, Ramirez, D, 11/1/1991)

(15, PP123456, PP, Kim, Ramirez, S, 1/5/1990)

5. **reservation(reservation\_number, passenger\_id, ticket\_id,** **user\_id, reserved\_datetime, ticket\_price, reservation\_status[cancelled|checked-in|onboarded]):** User can reserve a ticket for a flight or as many tickets as if with different passenger information. The reservation can be used as user’s flight history when the flight status (from flight relation) is done. Ticket price and ticket class are saved as static at the time user reversed a ticket. When the flight status is ready, onboarded status is used to determine whether a passenger is present.

(ABC1231, 1, 1, M1232, 10/06/2019, 1000, NULL)

(ABC1232, 1, 1, M1234, 10/06/2019, 1200, checked-in)

(ABC1233, 1, 1, M1231, 10/06/2019, 1000, checked-in)

(ABC1234, 1, 1, M1230, 10/06/2019, 1100, NULL)

(ABC1235, 1, 1, M1235, 10/06/2019, 1000, NULL)

(ABC1236, 1, 1, M1236, 10/06/2019, 2000, NULL)

(ABC1237, 1, 1, M1237, 10/06/2019, 100000, NULL)

(ABC1238, 1, 1, M1238, 10/06/2019, 1000, NULL)

(ABC1239, 1, 1, M1239, 10/06/2019, 1000, checked-in)

(ABC1230, 1, 1, M1240, 10/06/2019, 1000, NULL)

(ABC1240, 1, 1, M1241, 10/06/2019, 1000, NULL)

(ABC1241, 1, 1, M1242, 10/06/2019, 1000, NULL)

(ABC1242, 1, 1, M1243, 10/06/2019, 1000, NULL)

(ABC1243, 1, 1, M1255, 10/06/2019, 1000, NULL)

(ABC1244, 1, 1, M1999, 10/06/2019, 1000, NULL)

(ABC1245, 1, 1, M1111, 10/06/2019, 1000, NULL)

(ABC1246, 1, 1, M1112, 10/06/2019, 1000, NULL)

6. **flight(flight\_id, plane\_id, airport\_id, departure\_datetime, arrival\_datetime, depart\_from, arrive\_to, flight\_service [multiple], flight\_status [onboarding|delay|postponed|ready|departed|arrived], gate\_number):** When a flight is generated by System with a unique id. Each flight is of each plane, but a plane can be assigned to as many flights as if with different departure time and different airport, and next departure time cannot be before its own arrival time of the current flight. Each flight has information of where it departs from and where it arrives to. Each flight is provided with some available services (wifi, handicap, ...). Service can be one or multiple and saved as static in one attribute. Also, each flight has its own status, i.e: to let passengers know whether it is going to fly on-time or postponed, ready status is to determine when all passengers at the gate are onboarded. Gate number is assigned to let passengers know onto where they will be boarding

(1, 919304, 300,  May 29 8:00, May 20 9:00, San Jose, Las Vegas, normal, 1)

(113304, 80,  May 21 8: 00, May 20 9:00,  Las Vegas, San Jose, needed, 2)

(3, 929301, 350, May 22 8:00, May 20 9:00,  Las Vegas, San Jose, needed, 2)

(519404, 150,  May 25 8:00, May 20 9:00,  Las Vegas, San Jose, needed, 2)

(222332, 300,  May 27 8:00, May 20 9:00,  Las Vegas, San Jose, needed, 2)

(592013, 80,  May 20 8:00, May 20 9:00,  Las Vegas, San Jose, needed, 2)

(590192, 300, Apr 10, 8:00, May 20 9:00,  San Jose, New York, needed, 2)

(482901, 150, repairing, Apr 11, 8:00, May 20 9:00,  San Jose, New York, needed, 2)

(492012, 300, OK,  Apr 12, 8:00, May 20 9:00,  San Jose, New York, needed, 2)

(919323, 200, repairing,  Apr 13, 8:00, May 20 9:00,  San Jose, New York, needed, 2)

(10, 300, repairing,  Apr 14, 8:00, May 20 9:00,  San Jose, New York, needed, 2)

(11, 150, OK,  Apr 15, 8:00, May 20 9:00,  San Jose, New York, needed, 2)

(12, 300, OK,  Apr 16, 8:00, May 20 9:00,  San Jose, New York, needed, 2)

(13, 300, repairing,  Apr 17, 8:00, May 20 9:00,  San Jose, New York, needed, 2)

(14, 200, repairing,  Apr 18, 8:00, May 20 9:00,  San Jose, New York, needed, 2)

(15, 300, OK,  Apr 19, 8:00, May 20 9:00,  San Jose, New York, needed, 2)

7. **gate(gate\_number, gate\_status [occupied|available])**: Gate number is assigned by System, and status to let System know whether gate is available for a flight to assign to

(1, occupied)

(2, available)

(3, occupied)

(4, available)

(5, available)

(6, available)

(7, occupied)

(8, occupied)

(9, available)

(10, available)

(11, occupied)

(12, occupied)

(13, occupied)

(14, available)

(15, occupied)

8. **plane(plane\_id, seating\_capacity, maintenance\_status[OK|repairing]):** Each plane will be generated by System with an auto-increment integer as plane id, has its own seating capacity to decide the number of tickets will be generated for an associated flight, and has maintenance status to let System know whether to assign a flight to the plane

(919304, 300, OK)

(113304, 80, repairing)

(329304, 200, OK)

(519404, 150, OK)

(222332, 300, OK)

(592013, 80, repairing)

(590192, 300, repairing)

(482901, 150, repairing)

(492012, 300, OK)

(919323, 200, repairing)

(10, 300, repairing)

(11, 150, OK)

(12, 300, OK)

(13, 300, repairing)

(14, 200, repairing)

(15, 300, OK)