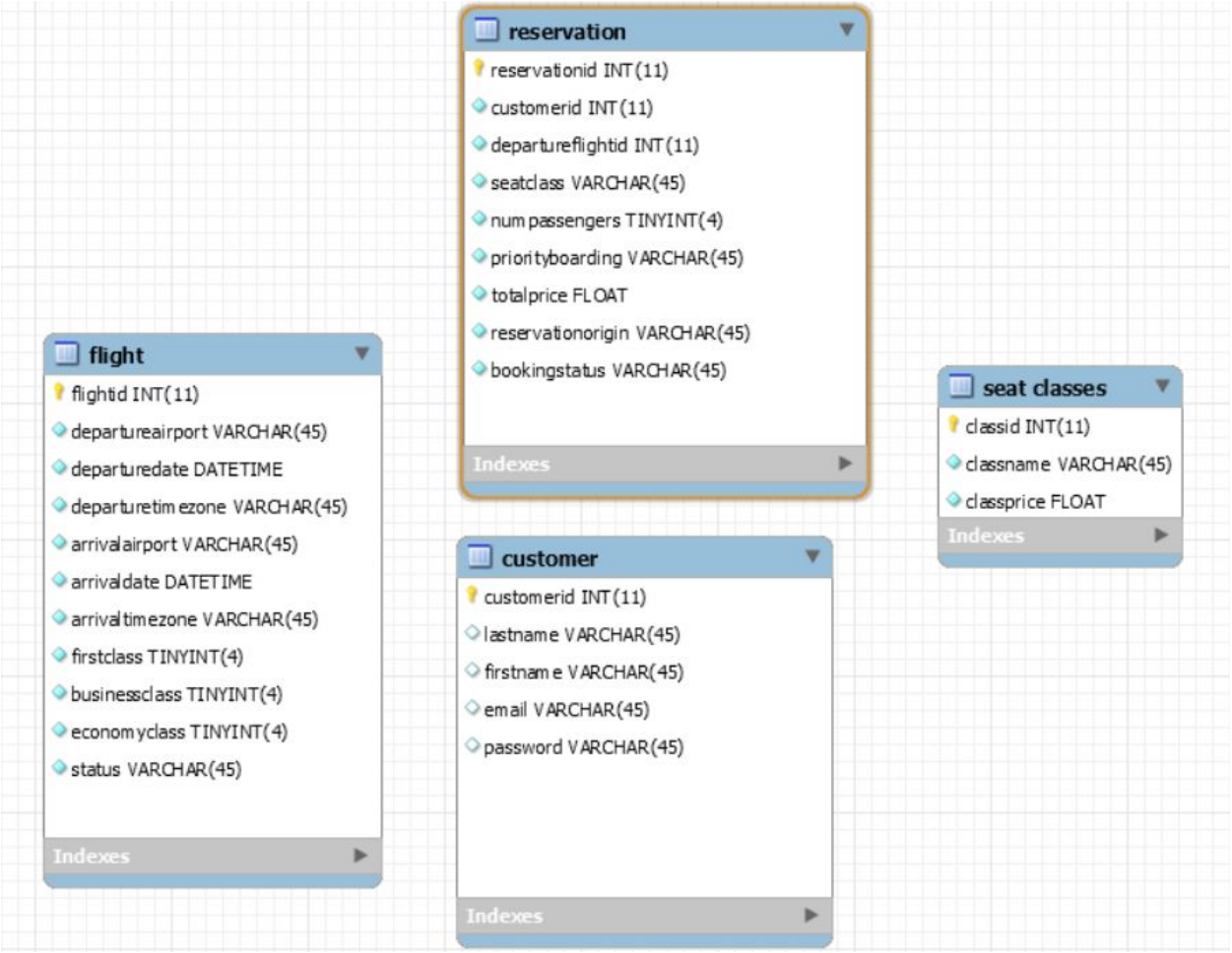
Andrew Bell, Nicholas Hinds, Andrew Terrado
CST 438 Group Project- Team 2
Kana Flights Database Design and Major Classes

Overall Database Diagram



The database was designed to encapsulate core data into 4 distinct tables: flight, reservation, customer, and seat classes. Each table contains specific and mostly unique information, with minimal overlap to adjacent tables. This was done to keep the tables separate, organized, and provide an efficient and logical path to information. For added simplicity, our reservation table handles both the website and API generated reservations. In order to differentiate between the two reservation sources, the reservation origin is recorded.

Database Breakdown

Flight

-This table contains information regarding all of our available flights. This includes airport locations, flight time, and available seating capacities

	Description	**Example**
int flightid	Flight identification number	1, 2, 3
varhchar departureairport	Departure airport location	Monterey
DateTime departuredate	Date/time of departure	2020-09-01 20:00:00
varhchar departuretimezone	Time zone of departure airport	PST
varhchar arrivalairport	Destination airport location	San Diego
DateTime arrivaldate	Date/time of arrival	2020-09-01 20:00:00
varhchar arrivaltimezone	Time zone of destination airport	HST
tinyint firstclass	First Class Seat Capacity	1,2,3, etc.
tinyint businessclass	Business Class Seat Capacity	1,2,3, etc.
tinyint economyclass	Economy Class Seat Capacity	1,2,3, etc.
varchar status	Status of flight	On-time, delayed

ie:

flightid	departureairport	departuredate	departuretimezone	arrivalairport	arrivaldate	arrivaltimezone	firstclass	businessclass	economyclass	status
1	Honolulu	2020-09-01 08:00:00	HST	Monterey	2020-09-01 16:00:00	PST	20	30	40	on time
2	Honolulu	2020-09-01 10:00:00	HST	San Diego	2020-09-01 18:00:00	PST	10	20	30	on time

Customer

-The Customer table contains user information, such as name, email, and password.

	Description	**Example**
int customerid	Customer identification number	1,2,3
varhchar lastname	Customer last name	Rey
varhchar firstname	Customer first name	Monte
varhchar email	Customer email address	kana@kana.com
varhchar password	Customer password	otterotter

ie:

customerid	lastname	firstname	email	password
1	Gates	Bill	bill@microsoft.com	windows

Reservation

-The reservation table contains entries for booked reservations. This table includes total price, number of passengers, etc. It references the flight and customer tables though the flightid and customerid respectively.

	Description	**Example**
int reservationid	Reservation identification number	1, 2, 3
int customerid	Customer identification number	1, 2, 3
int departureflightid	Flight identification number	1, 2, 3
varchar seatclass	Seating class for reservation	first, business
Tinyint numpassengers	Number of passengers booked	1, 2, 3
varchar priorityboarding	Priority boarding status	yes, no
Float totalprice	Total price of reservation	300, 400
varchar reservationorigin	Origin of reservation(site vs API)	kana, planner
varchar bookingstatus	Status of reservation	confirmed, cancelled

ie:

reservationid	customerid	departureflightid	seatclass	numpassengers	priorityboarding	totalprice	reservationorigin	bookingstatus
1	3	1	first	1	no	300	kana	confirmed
2	2	5	business	2	no	400	planner	cancelled

Seat Classes
-Planned to be used for data validation. It contains seat class names and prices.

	Description	Example
int classid	Class identification number	1, 2, 3
varchar classname	Class name	first, economy, business
Float classprice	Price for a seat in certain class	100, 200, 300

ie:

classid	classname	classprice
1	first	300
2	business	200

Major Classes and Components

FlightService Class

-Responsible for searching for available flights and booking new reservations

Methods

public List<Flight> getFlightInfoArrival(String arrivalairport)
-Takes destination airport string and returns list of all flights that feature the given destination

public List<Flight> getFlightInfoDeparture(String departureairport)
-Takes departure airport string and returns list of all flights that feature the given departure airport

public List<Flight> getAvailableFights(String departureAirport, String arrivalAirport, String dateStr)
-Takes departure airport, arrival airport, and date of flight, returns a list of flights that match the parameter.

public Reservation requestReservation(String email, String seatClass, int numPassengers, boolean priorityBoarding, String origin, int flightID)
-Takes email, seat class, number of passengers, boolean of priority boarding, origin of reservation, and the flight ID. Creates a reservation entry with given parameters and a new customer entry if the email is not in the database.

CustomerService Class

-Responsible for finding previous reservation information. Also allows users to cancel flights.

Methods

public List<FlightInfo> getPreviousFlights(String email)

-Takes a user's email and creates a list of flights that the user has reservations associated with

public List<ReservationFlightInfo> getPreviousReservations(String email)

public List<ReservationFlightInfo> getPreviousReservationsRest(String email)

-Standard and Rest version. This takes the user's email and generates a special data list that contains information about a user's reservations and the flight associated with the reservation. Separate versions to prevent API from accessing non-planner reservations.

public void updateStatus(String flightToCancel)

public void updateStatusRest(String flightToCancel)

-Standard and rest version. Takes user reservationID, ie: flightToCancel. Finds flight by reservation number and changes its booking status to "cancelled". Separate versions to prevent API from accessing Kana non-planner reservations.

FlightRestController

Methods

```
@GetMapping("/api/flights/arrival/{arrivalAirport}")
   public ResponseEntity<List<Flight>> getFlightArrival(
         @PathVariable("arrivalAirport") String arrivalAirport)
```

-Takes destination(arrival) airport string and returns a list of all flights going to this destination

```
 @GetMapping("/api/flights/departure/{departureAirport}")
   public ResponseEntity<List<Flight>> getFlightDeparture(
         @PathVariable("departureAirport") String departureAirport)
```

-Takes departure airport string and returns a list of all flights leaving from this airport

```
 @GetMapping("/api/flights/{departureAirport}/{arrivalAirport}")
   public ResponseEntity<List<Flight>> getAvailableFlights(
         @PathVariable("departureAirport") String departureAirport,
         @PathVariable("arrivalAirport") String arrivalAirport,
         String date
   )
```

-Takes departure airport, destination airport, and date, and returns lists of flights that match the given parameters.

```
  @GetMapping("/api/flights2/{departureAirport}/{arrivalAirport}/{month}/{day}/{year}")
   public ResponseEntity<List<Flight>> getAvailableFlights(
         @PathVariable("departureAirport") String departureAirport,
         @PathVariable("arrivalAirport") String arrivalAirport,
```

```
        @PathVariable("month") String month,
        @PathVariable("day") String day,
        @PathVariable("year") String year
    )
```
-Takes departure airport, destination airport, month, day, and year. Returns lists of flights that match the given parameters.

```
    @PostMapping("/api/flights")
    public ResponseEntity<List<Flight>> getAvailableFlightsPost(
        @RequestParam("departureAirport") String departureAirport,
        @RequestParam("arrivalAirport") String arrivalAirport,
        @RequestParam("date") String date
    )
```
-Takes departure airport, destination airport, and date, and returns lists of flights that match the given parameters.

```
    @PostMapping("/api/flight/reservation")
    public ResponseEntity<Reservation> createReservation(
        @RequestParam("email") String email,
        @RequestParam("seatClass") String seatClass,
        @RequestParam("numPassengers") int numPassengers,
        @RequestParam("prioBoarding") boolean prioBoarding,
        @RequestParam("flightID") int flightID
    )
```
-Takes email, seat class, number of passengers, priority boarding status, flight ID number. Generates reservation entry in reservation database and records given parameters.

```
@GetMapping("/api/flight/reservation2/{email}/{seatClass}/{numPassengers}/{prioBoarding}/{flig
htID}")
    public ResponseEntity<Reservation> createReservation2(
        @PathVariable("email") String email,
        @PathVariable("seatClass") String seatClass,
        @PathVariable("numPassengers") int numPassengers,
        @PathVariable("prioBoarding") boolean prioBoarding,
        @PathVariable("flightID") int flightID
    )
```
-Takes email, seat class, number of passengers, priority boarding status, flight ID number. Generates reservation entry in reservation database and records given parameters.

CustomerRestController

Methods

public ResponseEntity<List<FlightInfo>> getCustomerPreviousFlights(
@PathVariable("email") String email)
-Takes a user’s email and creates a list of flights that the user has reservations associated wit

public ResponseEntity<List<ReservationFlightInfo>> getCustomerPreviousReservationsRest(
@PathVariable("email") String email)
-Takes a user’s email and creates a list of the user's reservations.

public ResponseEntity<Reservation> cancelReservationRest(
@PathVariable("reservationID") Integer reservationID)

Attached below is the API documentation that we provided to the other teams. It goes into much more detail about the API functionality and provides examples of expected outcomes. The table of contents was modified to match the page numbers here.

Andrew Bell, Nicholas Hinds, Andrew Terrado
CST 438 Group Project

Flights API Documentation (v3.03), 8/13/2020

Table of Contents

Changelog (v3.03)

- Date implementation bug fix, revised successful output for getMapping (flight/date)

API Summary

https://kana-flight-service.herokuapp.com/

API Basic Workflow

1. **Search for flights** with location names, date, etc. to get flight ID,count of seats available, etc.
2. **Book reservations** with email, seat class choice, passenger count, priority boarding status, and the flight ID. Return info includes reservationID
3. **Look up existing reservation** information with the user's email address.
4. **Cancel flights** with the reservationID

Need flight information?

@GetMapping /api/flights/departure/{departureAirport}
@GetMapping /api/flights/arrival/{arrivalAirport}
@GetMapping /api/flights/{departureAirport}/{arrivalAirport}
@GetMapping /api/flights2/{departureAirport}/{arrivalAirport}/{month}/{day}/{year}

Post Option

```
@PostMapping("/api/flights")
        @RequestParam("departureAirport")
        @RequestParam("arrivalAirport")
        @RequestParam("date")
```

Need to book a reservation?

@GetMapping /api/flight/reservation2/{email}/{seatClass}/{numPassengers}/{prioBoarding}/{flightID}

Post Option

```
@PostMapping("/api/flight/reservation")
        @RequestParam("email")
        @RequestParam("seatClass")
        @RequestParam("numPassengers")
        @RequestParam("prioBoarding")
        @RequestParam("flightID")
```

Need to look up a reservation?

@GetMapping /api/previous_reservation/{email}

Need to cancel a reservation?

@GetMapping**:** api/cancel_reservation/{reservationID}

***Database is populated with flights to and from each location. Flights are available on 9/1 to 9/3, 2020**

Terms Reference Guide (JSON casing may differ)

Term	Examples	Notes
arrivalAirport	Monterey, Honolulu, San Diego	Destination airport
departureAirport	Monterey, Honolulu, San Diego	Departure airport
email	kana@kana.com	User email stored/will be stored on Kana database
seatClass	economy, business, first	Selected seat class
prioBoarding	true, false	Priority boarding status
flightID	1, 2,3, etc	ID of flight
numPassengers	1, 2, 3, etc.	Number of passengers booked
firstclass	1, 2, 3, etc.	Number of available class seats
businessclass	1, 2, 3, etc.	Number of available class seats
economyclass	1, 2, 3, etc.	Number of available class seats
reservationorigin	kana, planner	Origin of reservation
totalPrice	1000.0, etc.	Total price of reservation
bookingStatus	Confirmed, cancelled	Status of reservation
status	on time, delayed	Status of flight
customerid	1, 2, 3, etc.	Customer ID in Kana database
reservationid	1, 2, 3, etc.	Reservation ID in Kana database
month	09, 10, 11, etc.	Numeric representation: month
day	01, 02, 03, etc.	Numeric representation: day
year	2020	Numeric representation: year
date	09/01/2020	Numeric representation: date

Pricing Reference Guide (Pricing handled automatically by Kana services)
First- $300, Business- $200, economy- $100. Priority Boarding- $100 per passenger

Searching Flights

- **flightid will be necessary for the reservation booking process**

Search For All Flights From A Specific Departure Airport
@GetMapping /api/flights/departure/{departureAirport}

ie: /api/flights/departure/San Diego
Successful Output Example:


```
{
    "flightid": 5,
    "departureairport": "San Diego",
    "arrivalairport": "Honolulu",
    "departuredate":
"2020-09-01T15:00:00.000+00:00",
    "firstclass": 0,
    "businessclass": 30,
    "economyclass": 40,
    "status": "on time"
  },
  {
    "flightid": 6,
    "departureairport": "San Diego",
    "arrivalairport": "Monterey",
    "departuredate":
"2020-09-01T17:00:00.000+00:00",
    "firstclass": 10,
    "businessclass": 19,
    "economyclass": 30,
    "status": "on time"
  },
```


Potential Causes For Unsuccessful Output:

- Incorrect input

Unsuccessful Output Example:
(empty null response)

Search For All Flights To A Specific Destination Airport
@GetMapping /api/flights/arrival/{arrivalAirport}

ie: /api/flights/arrival/Monterey
Successful Output Example


```
    "flightid": 1,
    "departureairport": "Honolulu",
    "arrivalairport": "Monterey",
    "departuredate":
"2020-09-01T15:00:00.000+00:00",
    "firstclass": 0,
    "businessclass": 30,
    "economyclass": 40,
    "status": "on time"
  },
  {
    "flightid": 3,
    "departureairport": "Honolulu",
    "arrivalairport": "Monterey",
    "departuredate":
"2020-09-01T19:00:00.000+00:00",
    "firstclass": 5,
    "businessclass": 10,
    "economyclass": 20,
    "status": "on time"
  },
```

Potential Causes For Unsuccessful Output:

- Incorrect input

Unsuccessful Output Example:
(empty null response)

Search For All Flights Between Two Locations (Departure to Destination)- Get/Post
@GetMapping /api/flights/{departureAirport}/{arrivalAirport}

ie: /api/flights/Honolulu/Monterey
Successful Output Example:


```
[
  {
    "flightid": 1,
    "departureairport": "Honolulu",
    "arrivalairport": "Monterey",
    "departuredate":
"2020-09-01T15:00:00.000+00:00",
    "firstclass": 0,
    "businessclass": 30,
    "economyclass": 40,
    "status": "on time"
  },
  {
    "flightid": 3,
    "departureairport": "Honolulu",
    "arrivalairport": "Monterey",
    "departuredate":
"2020-09-01T19:00:00.000+00:00",
    "firstclass": 5,
    "businessclass": 10,
    "economyclass": 20,
    "status": "on time"
  }
]
```


Potential Causes For Unsuccessful Output:

- Incorrect input

Unsuccessful Output Example:
[ ]

Post Option

- ***Deprecated: For current version of this postMapping with date parameter, see next page***
- ***This version of the mapping is still functional***

@PostMapping("/api/flights")
@RequestParam("departureAirport")
@RequestParam("arrivalAirport")

Search Flights Between Two Locations (Departure to Destination:Specific Date)- Get/Post

- getMapping date is separated into month, day, and year path variables

@GetMapping /api/flights2/{departureAirport}/{arrivalAirport}/{month}/{day}/{year}

ie: /api/flights2/Honolulu/San Diego/09/01/2020
Successful Output Example:


```
[
  {
    "flightid": 2,
    "departureairport": "Honolulu",
    "arrivalairport": "San Diego",
    "departuredate":
"2020-09-01T17:00:00.000+00:00",
    "firstclass": 10,
    "businessclass": 20,
    "economyclass": 30,
    "status": "on time"
  },
  {
    "flightid": 4,
    "departureairport": "Honolulu",
    "arrivalairport": "San Diego",
    "departuredate":
"2020-09-01T21:00:00.000+00:00",
    "firstclass": 0,
    "businessclass": 5,
    "economyclass": 15,
    "status": "on time"
  }
]
```


Potential Causes For Unsuccessful Output:

- Incorrect input

Unsuccessful Output Example:
[ ]

Post Mapping Option

- postMapping uses a conventional date string, ie: “09/01/2020”

@PostMapping("/api/flights")
@RequestParam("departureAirport")
@RequestParam("arrivalAirport")
@RequestParam("date")

Booking Reservations

- **flightid will be necessary for the reservation booking process**

Book Reservation - Get or Post Options

@GetMapping /api/flight/reservation2/{email}/{seatClass}/{numPassengers}/{prioBoarding}/{flightID}

ie: /api/flight/reservation2/bill@microsoft.com/first/20/true/1

Successful Output Example:


```
{
   "reservationid": 15,
   "customerid": 1,
   "departureflightid": 1,
   "seatclass": "first",
   "numpassengers": 20,
   "priorityboarding": "yes",
   "totalprice": 8000.0,
   "reservationorigin": "planner",
   "bookingStatus": "confirmed"
}
```


Potential Causes For Unsuccessful Output:

- Too many seats booked
- Incorrect input

Unsuccessful Output Example :


```
"timestamp": "2020-08-11T20:17:31.015+00:00",
   "status": 500,
   "error": "Internal Server Error",
```


Post Mapping Option

@PostMapping("/api/flight/reservation")
@RequestParam("email")
@RequestParam("seatClass")
@RequestParam("numPassengers")
@RequestParam("prioBoarding")
@RequestParam("flightID")

Reviewing Booked Flight Information

- Shows active reservations that have not been cancelled
- Only shows reservations booked by the travel planner
- Returns reservation information

@GetMapping /api/previous_reservation/{email}

ie: api/previous_reservation/bill@microsoft.com

Successful Output Example:


```
[
   {
      "reservationId": 5,
      "id": 11,
      "departureAirport": "Monterey",
      "arrivalAirport": "San Diego",
      "departureDate": "2020-09-01T19:00:00.000+00:00",
      "status": "on time",
      "bookingStatus": "confirmed",
      "reservationOrigin": "planner"
   }
]
```


Potential Causes For Unsuccessful Output:

- Incorrect email input

Unsuccessful Output Example:


```
"timestamp": "2020-08-11T20:17:31.015+00:00",
   "status": 500,
   "error": "Internal Server Error
```


Cancelling Flights

@GetMapping**:** api/cancel_reservation/{reservationID}

ie: api/cancel_reservation/

Successful Output Example (bookingStatus changed to cancelled)


```
{
   "reservationid": 5,
   "customerid": 1,
   "departureflightid": 11,
   "seatclass": "economy",
   "numpassengers": 2,
   "priorityboarding": "yes",
   "totalprice": 600.0,
   "reservationorigin": "planner",
   "bookingStatus": "cancelled"
}
```


Potential Causes For Unsuccessful Output:

- Incorrect email input
- Planner does not have access to a reservation (ie: reservations booked on Kana site)

Unsuccessful Output Example:

(empty null response)