Air Travel Exercise

Entity-Relationship Model

Airline

has-many: aircraft

has-many: airports (i.e. places they can fly to and from)

properties: name (string), address (string), total net revenue(integer), login credentials

Airport

This table will be static.

has-many: Distance

Property: name (string)

Distance

This table will be static.

belongs-to: Origin Airport

belongs-to: Destination Airport

Property: Distance (integer)

Aircraft Type

This table will be static.

Properties: first-class capacity (integer), business-class capacity (integer), economy-class capacity (integer), range (integer), manufacturer (string), aircraft type(string), cost per mile (integer)

Aircraft

has-many: journeys

has-one: aircraft type

belongs-to: airline

Properties: FAA registration code (string, unique for each aircraft),

Journey

belongs-to: aircraft

belongs-to: ticket

belongs-to: schedule

belongs-to: journey log

has-one: origin airport

has-one: destination airport

Properties: departure date (date), arrival date (date), sold first-class tickets (integer), sold business-class tickets (integer), sold economy-class tickets (integer)

(Each journey has a revenue, but this can be derived from its other properties).

Journey Log

(This contains journeys which have been made in the past).

has-many: journeys

Schedule

(This contains journeys which have not yet been made).

has-many: journeys

Ticket

belongs-to: passenger

has-one: journey

Properties: class (integer, one of first, business or economy), cost (integer)

Passenger

has-many: tickets

Properties: name (string), address (string), total distance flown (integer), total money spent (integer), login credentials

Workflow

Airports are set up at the start, and distances are set up between airports. Available aircraft types will be defined. These should be done using a seed file since these tables are static.

The will be a Welcome Page, which explains that the objective of the application. This page will allow users to:

- Set themselves up as airlines (new users)
- Log in as airlines (existing users)
- Set themselves up as passengers (new users)
- Log in as passengers (existing users)

The pages for new users will set up login credentials for the passenger and airline, and explain what they can do once they are registered. (Airlines and passengers are equivalent to what most applications call administrators and users).

As part of the registration process for airlines, the airline user enters the aircraft the airline will own. A registered airline can own several aircraft from the available types. We are not putting restrictions on the number of aircraft each airline can own, the cost of buying the aircraft, or which airports an airline can fly to. Airlines cannot acquire (or lose) aircraft after the registration process has finished.

Airlines enter journeys into the schedule, using the aircraft they have available. Aircraft can only fly a journey if:

1. the aircraft has sufficient range to make the journey; and

2. the origin of one journey is the destination of the aircraft's previous journey.

Each aircraft is initialized with a dummy journey to give it an origin for its first real journey. There should be a query which returns where each aircraft is scheduled to be on any given date, which validates whether the airline is trying to enter a journey from an unacceptable airport. Aircraft are allowed to arrive at and airport and depart from it on the same day (we are only considering days in the first instance, not hours. A possible extension would be to use hours as well as days, and give each aircraft type a "turnover time" between when it can arrive and depart from the same airport.

For each journey in the schedule, tickets can be sold to passengers. Once the start date of a scheduled flight is reached, the flight is removed from the schedule and added to the journey log. During the move of the journey from the schedule to the journey log, the total revenue from the journey is added to the airline's total revenue. The revenue from each flight is found by subtracting the cost of the journey (range x cost per mile) from the income from all tickets sold for the journey. It is possible for a journey to make a loss (negative revenue) if it does not sell enough tickets. Journeys cannot be withdrawn from the schedule.

The internal method call which moves journeys from the schedule to the journey log is made as part of the start-up or login process. This method simply looks for journeys in the schedule with a departure date earlier than the current date, and moves any such journeys to the journey log. This only needs to be done once per login.

Passengers must register to be set up on the system, during which they enter their name and address. They then enter:

- their desired origin and destination;

- first acceptable date of departure;

- last acceptable date of departure.

(We are not considering return tickets in the first instance). Passengers are given a list of journeys in the schedule which acceptable departure dates. The can select the flight and find out how many tickets in each class are available, together with the price per ticket. They can then purchase as many tickets as they wish, provided the aircraft still has capacity. (We assume passengers have infinite money to buy tickets). Passengers can select as many tickets as they wish from any scheduled journey, in any class.

There will be an authorized page for existing passengers, which allows the passenger to view tickets which they have previously purchased (most recent first with paging).

The journey log can be used to calculate how much revenue each airline has made. There will be an authorized page for existing airlines, which lists the flights that airline has made in chronological order (most recent first with paging) and the current net revenue of the airline.

External APIs

Devise	For registering new users and authenticating logins
Pagination	For pagination the summary pages for passengers and airlines (a lot of APIs to choose from here, Pagination looks easy but I may choose another one)