CSC/ECE 517 (OO Design and Development)

Program 2: Ruby on Rails

Teams may consist of 2 or 3 members

Submission Due Date - February 26, 2021 (Friday) 11:59 PM

Resubmission Due Date - March 03, 2021 (Wednesday) 11:59 PM

# Flight Reservation System

You are tasked with creating a flight reservation system for an airline, say Divided Airlines (choose your airline name!). This system allows the *User* to view *Flights* and make *Reservations.* They can alsoprovide *Feedback* for the same. It also allows the *Admin* to create *Flights* and view information on all *Users, Reservation* and *Feedback* .

[There will be 5 main components in the system](https://docs.google.com/document/d/131xL_WKAlh9Pf7uj9xC9PcX2jcFv-Kcb_sFZ_MdjdyY/edit#heading=h.rqa6864nh5gh):

1. Admin
2. User
3. Flight
4. Reservation
5. Feedback

## Admin

[The system should have only one preconfigured admin](https://docs.google.com/document/d/131xL_WKAlh9Pf7uj9xC9PcX2jcFv-Kcb_sFZ_MdjdyY/edit#heading=h.edzs897m9h6f) with at least the following attributes:

* Name
* Email
* Password
* Phone number
* Address
* Credit card information (Fake one!)
  + Name on card
  + 16-digit card number
  + Expiration date (mm/yy)
  + 3-digit cvv (security code)

Admin should be able to:

* Log in with an email and password.
* Edit her/his own profile - should not be able to update email and password.
* Admin should *not* be able to delete the admin account
* Create/view/edit/delete users.
* Create/view/edit/delete flights.
* Create/view/edit/delete reservations.
* Create/view/edit/delete feedback.
* View a list of flights (previous and upcoming) and the corresponding list of users who have made reservations for the same.
* View all feedback given by users who have made reservations for flights they have travelled on. Feedback should be viewed grouped on flights, not users.
* An admin should also be capable of performing all operations performed by the Users mentioned below.

## Flight

The system should have *Flights* scheduled which the *Admin* will feed into the database. The *User* would be able to select one of these flights according to their preferences. This class will have at least the following attributes:

* Flight Number
* Origin airport
* Destination airport
* Departure date
* Departure Time
* Flight price
* Flight Capacity (Maximum no.of seats available)

## Reservation

The system should have a *Reservation* made when a *User* books a *Flight*. The functionalities related to reservations are mentioned under *User* and *Admin*. This class will have at least the following attributes:

* User ID
* Flight ID
* Confirmation Number
* Class (for example, economy class, business class, first class)

## User

[The system should have users who register themselves to book a flight](https://docs.google.com/document/d/131xL_WKAlh9Pf7uj9xC9PcX2jcFv-Kcb_sFZ_MdjdyY/edit#heading=h.1rx00irjgaby) and have at least the following attributes:

* Name
* Email
* Password
* Phone number
* Address
* Credit card information (Fake ones!)
  + Name on card
  + 16-digit card number
  + Expiration date (mm/yy)
  + 3-digit cvv (security code)

[Users should be able to](https://docs.google.com/document/d/131xL_WKAlh9Pf7uj9xC9PcX2jcFv-Kcb_sFZ_MdjdyY/edit#heading=h.i24mlwh0ab9):

* Log in with an email and password.
* Edit her/his own profile.
* View available flights based on their preferences (like airports and departure dates).
* Reserve a round-trip or a one-way flight and choose a flying class that they prefer.
* Modify the departure/return dates for a flight reservation. The origin and destination airports cannot be modified once the reservation is made.
* Delete a flight reservation.
* View the flight history for their account.
* View the upcoming flight(s), if any.
* Give feedback to the flights on which they have traveled. The user cannot give feedback on any upcoming flight(s) or flight(s) they never booked.
* Should receive an email with the reservation confirmation number after booking a flight.

## Feedback

The system should have *Users* who can give *Feedback* about the flights they have travelled on, the reservation process, the website, etc. Note that it can be given only after their journey is complete i.e. only to previous flights, and not upcoming flights. It should have at least the following attributes:

* User ID
* Flight ID
* Feedback Response

## **Bonus (Extra Credit)**

You can do any or all of the below for extra credit (each item below is worth 5 points).

* No user should be able to access any private content associated with another user/admin's account. For example, a user should not have the access to edit any flight information.
* All users can use their LinkedIn/Google/Facebook account (implementing any one of them is enough) to log into the system.

# **Frequently Asked Questions (FAQs)**

* How to start with this project?
  + Scaffolding is a great way to create the initial structure of this project. It automatically creates many files and basic CRUD operations for you. You can go through [this link](https://www.rubyguides.com/2020/03/rails-scaffolding/) to get more information on it. There are several such resources available online.
* Can we generate more classes, if required?
  + The documentation guides through the basic entities and functionalities that are required. You are free to add more classes as per your design.
* Can we use any 3rd-party gems?
  + Yes, you can.
* If the admin account is predefined, how does the admin know how to log in? Do we just give the admin a predefined login and password?
  + Yes.
* Would a bare minimum UI consist of a page of links and simple HTML?
  + As long as the functionalities work, it is ok. UI is not given importance or points.
* Is the admin able to edit the existing information for users?
  + Yes. As mentioned above, an admin should be capable of performing all operations performed by the users.
* Admin has the right to create user accounts as well. When the admin creates an account for a user, the admin creates it with the user’s email address. Admin is also responsible for populating the list of flights that the users will view. So they can be responsible to set up a very basic database through the user interface so that the basic functionalities can be demonstrated.
* Is the extra credit included in 70 points for the program, or can we score more than 70?
  + You can score more than 70 if you finish all extra credit tasks.

# **Miscellaneous**

## **Ruby Version**

There is no requirement for a Ruby version. Anything **2.6.X and above** should work perfectly.

## **Repository**

* Please make sure your repository is **private** and is in the **ncsu.github.edu** domain.

## **Testing**

* Thoroughly test one model and one controller (prefer [RSpec](http://rspec.info) testing framework; see Week 5 online videos).

## **Deployment**

**Please ensure that your deployment is always accessible for grading.** You can deploy your app to any of the following:

* Heroku or any similar PaaS (OpenShift, etc.) with free plans ([Link to instructions](https://devcenter.heroku.com/articles/git))
* Amazon AWS
* NCSU VCL

Please deploy your application a couple of days before the deadline. This will give you a chance to work through any issues that arise. Be sure that it is active for two weeks after the deadline, so that grading can be completed.

# **Submission**

Your submission in Expertiza should consist of the following:

* A link to your deployed application
* A link to your repository (Keep the repository private for Round 1, this is just for our records)
* A README.md file containing:
  + Credentials for the preconfigured admin and any other information that reviewers would find useful
  + How to test various features (e.g., how to access certain pages, what details to enter in the form etc.)
  + How your program handles edge-case scenarios (e.g., what will happen if the admin deletes a flight which several users have booked?)