CSC/ECE 517 (OO Design and Development)
Program 2: Ruby on Rails

Teams may consist of 2 or 3 members
Submission Due Date - September 30, 2023 (Saturday) 11:59 PM
Resubmission Due Date - October 05, 2023 (Thursday) 11:59 PM

Rail Ticketing System

You are tasked with creating a railway ticket services system for a transportation company (choose your name). This system allows the *Passenger* to view *Trains* and buy *Tickets.* They can also write *Reviews* for the same. It also allows the *Admin* to create available *trains* and view information on all *Passengers, Tickets* and *Reviews*.

There will be 5 main components in the system:

1. Admin
2. Passenger
3. Train
4. Ticket
5. Review

You can add your own as long as the system works.

Admin

The system should have only one preconfigured admin with at least the following attributes:

- Username
- Name
- Email
- Password
- Phone number
- Address
- Credit number (Fake one!)

Admin should be able to:

- Log in with a username and password.
- Edit her/his own profile - should not be able to update ID, username, and password.
- Admin should not be able to delete the admin account (nor should anyone else).
- View all the trains that are available on the website.

- List trains by a specific departure/termination station.
- View all the users signed up for the website.
- List reviews written by a specific user (with name)
- List reviews written for a specific train (with train number)
- Create/view/edit/delete passengers.
- Create/view/edit/delete trains.
- Create/view/edit/delete tickets.
- Create/view/edit/delete reviews.

Train

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

- ID
- Train number
- Departure station
- Termination station
- Departure date
- Departure time
- Arrival date
- Arrival time
- Ticket price
- Train capacity (Maximum number of seats available)
- Number of seats left

Ticket

The system should have a *Ticket* made when a *Passenger* books a *Train*. The functionalities related to *Tickets* are mentioned under *Passenger* and *Admin*. This class will have at least the following attributes:

- ID
- Passenger ID
- Train ID
- Confirmation Number

We leave the decision to you regarding how to handle the ticket-buying process. Whether you want to have a separate transaction model is up to you.

Passenger

The system should have users who register themselves to book a train and have at least the following attributes:

- ID
- Name
- Email
- Password
- Phone number
- Address
- Credit-card information (Fake ones!)

Passengers should be able to:

- Sign up for a new account.
- Log in with a username and password.
- Edit her/his own profile, but should not be able to update their ID.
- Delete their own account.
- View all the trains that are available (only upcoming trains, not previous trains, and only trains that have available seats) in the system.
- List trains that have an average rating over a certain amount (e.g., average rating greater than 3).
- List trains by a specific departure/termination station.
- Book a train
- Check their own trip history
- Write reviews of the trains on which they have traveled. The user cannot give feedback on any train(s) they never booked.
- Edit the review he/she wrote, but should not be able to edit reviews that were written by other users.
- List reviews written by a specific user (with name)
- List reviews written for a specific train (with train number)
- Cancel a ticket.

Review

The system should have *Passengers* who can give *Reviews* about the trains they have traveled on, the reservation process, the website, etc. It should have at least the following attributes:

- Passenger ID
- Train ID
- Rating (1 – 5)
- Feedback

General Requirements

- There should be a link on the user's home page to let the user:
    - Edit his/her profile.
    - List his/her booking history.
    - List reviews are written by him/herself.

    - List all available trains on the website with their train number, departure station, termination station, departure date, departure time, arrival date, arrival time, ticket price, number of seats left, and average rating.
    - Search for reviews written by a specific user (with name).
    - Search for reviews written for a specific train (with train number).
- There should be a link on the admin's home page to let the admin:
    - Edit his/her profile.
    - List all users who have signed up for the website
    - List all trains on the website with their train number, departure station, termination station, departure date, departure time, arrival date, arrival time, ticket price, number of seats left, and average rating.
    - List all reviews written by all users.
- There should always be a way to let the user go back to the home page.
- Price should be automatically calculated when buying tickets.
- The number of seats left should be recalculated after each transaction.
- Rating should be recalculated after each review is submitted.
- If a user gets deleted, all reviews written by this user should be deleted as well.
- If a train gets deleted, all reviews written for this train should be deleted as well.
- There will be only one admin in the system, and the account is preconfigured. The admin account cannot be deleted.
- No other users should be able to access each other's profiles.
- No users should be able to edit another user's review.
- Ensure that your code performs appropriate validations (e.g., price, seat left cannot be negative, email address should be valid).
- Ensure the value for necessary fields is not empty before saving to the database.
- Make sure users are not able to access resources that they are not allowed to by simply changing the URL.
- In your README file, please document how to access certain pages in your app. Here are several examples:
    - By clicking what button on what page can a user buy a book?
    - By clicking what button on what page can a user can edit a review he/she has written?
    - By clicking what button on what page can an admin delete a user?

Bonus (Extra Credit)

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

- Implement a search function for the admin to use. The input is the train number; the search result is a list of users who booked this train.
- Implement a function to allow a user to buy a ticket for another user (the ticket can be viewed by both the user who pays for the ticket and the user who receives the ticket).

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 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. However, gems like Solidus should be avoided because this gem can generate an app for you and that is not allowed for the project.

- 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, you seed the database with this information and add to the README file.

- Would a bare minimum UI consist of a page of links and simple HTML?
    - As long as the functionalities work, it is ok.

- Is the admin able to edit the existing information for users?
    - Yes.

- Is the extra credit included in 100 points for the program, or can we score more than 100?
    - You can score more than 100 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. After the first round of review, you will be asked to make your repository public.
- And add all the TAs to be collaborators so your work can be graded.

Testing

- Thoroughly test **one** model and **one** controller (RSpec 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:

- PaaS (OpenShift, etc.) with free plans. Heroku does not work anymore.
- 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 it is active for two weeks after the deadline so that grading can be completed.

Please check if your website is UP and running at least once daily to avoid loss of points due to accidental mishaps.
While reviewing others' work, if you find that any website is down, please email them.
You can find the UnityID of a team member, from their github.ncsu.edu username.

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.).**