CSC/ECE 517 (OO Design and Development)

Program 2: Ruby on Rails

[Wed 02/15] Submission

[Fri 02/17] First feedback

[Tue 02/21] Resubmission

[Thu 02/23] Final review

**Note:**

As soon as you start working on the project, you must create a repo on [github.ncsu.edu](http://github.ncsu.edu/) and **add the course staff** and teammates as collaborators. Make sure to make your repository private to prevent potential plagiarism. Use the GitHub features you learned in Program 0 Github Introduction assignment, like Issues, Branches, Pull Requests, Project Board, etc.

# Online Book-shopping System

The task is to create an online book-shopping system using Ruby on Rails. It is similar to the practice you have in class, but with more components. Note that points will be assigned based on the functionality, **not the user interface.**

The main components of the system are:

1. Book
2. Transaction
3. Review
4. User
5. Admin

# Book

Books should have the following attributes **(required fields are indicated by an asterisk (“\*”)**:

* ID \*
* Name \*
* Author \*
* Publisher \*
* Price \*
* Stock \*

Transaction

Transactions should have the following attributes **(required fields are indicated by an asterisk (“\*”)**:

* Transaction number \*
* Book ID \*
* User ID \*
* Credit card number \*
* Address \*
* Phone number \*
* Quantity \*
* Total price \*

Review

Reviews should have the following attributes **(required fields are indicated by an asterisk (“\*”)**:

* ID \*
* Book ID \*
* User ID \*
* Rating (1 - 5) \*
* Review \*

# User

Users should have the following attributes **(required fields are indicated by an asterisk (“\*”)**:

* ID \*
* Username \*
* Password \*
* Name \*
* Email \*
* Address
* Credit card number
* Phone number

Users 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 books that are available on the website
* List books that have an average rating over a certain amount.
* List books written by a specific author.
* Buy a book or books
* Check their own purchase history (or transaction history)
* Write a review of a book
* 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 username)
* List reviews written for a specific book (with book name)

# Admin

Admin should have the following attributes **(required fields are indicated by an asterisk (“\*”)**:

* ID \*
* Username \*
* Password \*
* Name \*
* Email \*

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 books that are available on the website
* List books written by a specific author.
* View all the users signed up for the website
* List reviews written by a specific user (with username)
* List reviews written for a specific book (with book name)
* Create/view/edit/delete users.
* Create/view/edit/delete books.
* Create/view/edit/delete reviews.

# General Requirements

* There should be a link on the user’s home page to let the user:
  + Edit his/her profile.
  + List his/her shopping history.
  + List reviews are written by him/herself.
  + List all books on the website with their name, author, publisher, price, stock, and average rating.
  + Search for reviews written by a specific user (with username).
  + Search for reviews written for a specific book (with the book name).
* 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 books on the website with their name, author, publisher, price, stock, and average rating.
  + List all reviews written by all users.
* There should always be a way to let the user go back to the homepage.
* Price should be automatically calculated when buying books.
* Stock 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 book gets deleted, all reviews written for this book 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 profile.
* No users should be able to edit another user’s review.
* Ensure you have necessary validations (i.e., price, stock 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 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 a user can buy a book.
  + By clicking what button on what page a user can edit a review written by him/her.
  + By clicking what button on what page an admin can delete a user.

# Extra Credit:

Form pre-filling (3 points):

When a user tries to buy a book (create a transaction), the form is pre-filled with information in the user’s profile (credit card number, email, address, etc.).

Take care of race conditions (5 points):

When there is available stock, users should be able to buy a book. However, other users may try to buy the same book before a user checks out. This may lead to more books sold than the stock number. Handling this issue will earn 5 points of extra credit.

Implement a cart-like function (10 points):

Allow users to check out multiple different books at the same time. Instead of only allowing users to buy a certain amount of the same book, allow users to check out multiple different books at the same time.

# **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. However, gems like Solidus should be avoided.
* 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 it public.
* And add all the TAs to be collaborators so your work can be graded.

## **Testing**

* Thoroughly test **one** model and **one** controller ([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:

* PaaS (OpenShift, etc.) with free plans. Heroku does not work anymore.
* Amazon AWS
* [NCSU VCL](https://docs.google.com/document/d/168AveJMHh3trO2vWB9mQ0zGpI2VCZl3iB0g0a7pmjS0/edit?usp=sharing)

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.).**

# **Rubrics**

Here is the peer review rubric you will be reviewed on by your peers.

**Round 1:**

Coming soon