CSC/ECE 517 (OO Design and Development)

Program 2: Ruby on Rails

Teams may consist of 2 or 3 members

Submission Due Date - February 16, 2024 (Friday) 11:59 PM

Resubmission Due Date - February 22, 2024 (Thursday) 11:59 PM

You get 5 points for submitting your repo to Expertiza by 11:59 PM on Feb. 6. Empty repo counts as well.

# Event Management System You are assigned to develop an event management system for NCSU called WolfEvents. This system enables attendees to explore *Events* and purchase *Tickets*. *Attendees* can also provide *Reviews* for the events they have attended. Additionally, the *Admin* has the capability to create events, and access information on all *Attendees*, *Tickets*, and *Reviews*. There will be 6 main components in the system:

1. Admin
2. Attendee
3. Room
4. Event
5. Event Ticket
6. 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:

* Email
* Password
* Name
* Phone number
* Address
* Credit card information (Fake one!)

Admin should be able to:

* Log in with an email and a password.
* Edit her/his own profile - should not be able to update ID, email, and password.
* Admin should not be able to delete the admin account (nor should anyone else).
* View all the events that are available on the system.
* List events by a specific category, date, price (~ price range to be precise), and event name.
* View all the attendees signed up on the event management system.
* List reviews written by a specific attendee (with name)
* List reviews written for a specific event (with Event name)
* Create/view/edit/delete attendees.
* Create/view/edit/delete events.
* Create/view/edit/delete tickets.
* Create/view/edit/delete reviews.
* Create/view/edit/delete rooms.
  + Admin should be able to search only available rooms in a particular time slot  
      
    (If a room is already booked, that should not be visible for the admin as part of his/her event creation during the time slot when the room is booked)
* Admin should also be able to book events like an attendee and attend them.

## Event

The system should be initialized with scheduled *Events*, which only the *Admin* can create. The *Attendee* would be able to select one of these *Events* according to their preferences. This class should have, at a minimum, the following attributes.

* ID
* Event name
* Event room ID
* Event category (Concerts, Sports, Arts & Theatre, Miscellaneous/Family – Private)
* Event date
* Event start time
* Event end time
* Ticket price
* Number of seats left

We leave it to you on how consistency between seat capacity is managed between room and event. For example: When you are creating an event, you can get the room capacity and have an attribute in the Event component indicating it for further checks.

## Event Ticket

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

* ID
* Attendee ID
* Event ID
* Room 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.  
  
**Room**  
  
The system should allow the admin only to book events in a particular *Room* for a particular time slot. The functionalities related to *Room* are mentioned under *Admin*. This class will have at least the following attributes:

* ID
* Room location/address
* Room capacity

We leave the decision to you on managing consistency of capacity between the room and the event.

## Attendee

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

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

Attendees should be able to:

* Sign up for a new account.
* Log in with an email and a password.
* Edit their own profile, but should not be able to update their ID.
* Delete their own account.
  + All Dependents should be deleted. Ex: Tickets booked by attendee or reviews written for the attended events by attendee should be deleted.
* View all the events that are available in the system.
  + Condition 1: Only upcoming events should be visible.
  + Condition 2: Only events which are not sold out should be visible.
  + Condition 3: Filter Events by:
    - Category
    - Date
    - Price (~ price range to be precise)
  + Condition 4: Search Events by Event Name.
* Book an Event ticket
* Check their own event booking history.
  + At least the Event Name and Date should be shown, everything else is optional.
* Write reviews of the events that they attended.
  + Attendee should only be able to review events he/she attended
  + He/she should be able to review events of only Concerts, Sports, and Arts & Theatre categories. Miscellaneous/Private category events cannot be reviewed.
  + Attendees can review only after the event ends, and by logic, reviews should be visible only after the event ends.
* 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 Email of the user)
* List reviews written for a specific event (with Event Name)
* Cancel a ticket.
  + This would imply that the event gets its seats back.

## Review

The system should have *Attendees* who can give *Reviews* on the events they have attended, the booking process, the website, etc. It should have at least the following attributes:

* Attendee ID
* Event 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/her.
  + List all available events on the website with category, date, start time, end time, ticket price, event name, room ID, and number of seats left in the event.
  + Search for reviews written by a specific user (with the user’s email).
  + Search for reviews written for a specific event (with event name).
* There should be a link on the admin’s homepage to let the admin:
  + Edit his/her profile.
  + List all attendees who have signed up for the website
  + Create/Delete/Edit Events.
  + Create/Delete/Edit Rooms for events to be booked in them.
  + List all available events on the website with category, date, start time, end time, ticket price, event name, room ID, and number of seats left in the event.
  + List all reviews written by all attendees.
  + Search for reviews written by a specific user (with the user's email).
  + Search for reviews written for a specific event (with event name).
  + Admin should also be able to book tickets to attend an event.
* There should always be a way to let the user go back to the home page.
* Price should be automatically calculated when buying tickets.
  + You can just multiply the number of tickets purchased by each ticket price to get the total cost of ticket booking. You can ignore taxes or any other fee.
* The number of seats left should be recalculated after each transaction.
* If a user gets deleted, all reviews written by this user should be deleted as well.
* If a room gets deleted, all events booked in that particular room as well as all tickets for such events should be deleted as well.
* If an event gets deleted, reviews 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 user should be able to access another user’s profiles.
* No user should be able to edit another user’s review.
* You should ensure that your code performs appropriate validations (e.g., price, seat left cannot be negative, email address should be valid).
* You should ensure that the value for necessary fields is not empty before saving to the database.
* Users must not be 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 ticket?
  + By clicking what button on what page can a user edits 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 event name; the search result is a list of attendees who booked this event.
* Implement a function to allow an attendee to buy a ticket for another attendee (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. See [this Ruby Guide](https://www.rubyguides.com/2020/03/rails-scaffolding/) for more information. 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 it 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, i.e., can we score more than 100?
  + You can score more than 100 if you finish extra-credit tasks.
* Can I add other fields that are not in the requirements?
  + Yes! Feel free to implement all the features/attributes you want, on top of the necessary features/attributes mentioned in the doc.
* Do the 5 points we get for turning in early count as extra credit?  
  + No, they are part of the regular point value of the assignment.

# **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 **github.ncsu.edu** domain. After the first round of review, you will be asked to make your repository public. The first round of review is for functionality; the second round is for code style.
* And add all the TAs as 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 ([Instructions to deploy on 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.).**

# Grading Rubric

# Round 1 Rubric

| **Class and Database Design** |
| --- |
| 1. All required attributes for the admin are present and correctly configured. |
| 1. All required attributes for the attendees are present and correctly configured. |
| 1. The review contains all necessary attributes including ratings, feedback, etc. |
| 1. The event entry includes all essential attributes. |
| 1. The ticket is complete with all required attributes. |

| **Functionality** |
| --- |
| 1. Attendees can successfully sign up for a new account. |
| 1. Both attendees and admins can log in using a valid email and password. |
| 1. Attendees have the ability to edit and delete their own profiles. |
| 1. Both attendees and admins can view the complete list of events on the website. |
| 1. Attendees can filter out events based on category, date and price. |
| 1. Attendees can successfully purchase a ticket. |
| 1. Attendees can access and view their own event booking history. |
| 1. Attendees can write and edit their own event reviews but not those written by other attendees. |
| 1. Both attendees and admins can list reviews either by a specific attendee’s name or by a specific event name. |
| 1. Admins can edit their own profiles but cannot delete their admin accounts. |
| 1. Admins can view a list of all attendees who have signed up for the website. |
| 1. Admins have the capability to create, view, edit, and delete attendees, events, and reviews. |
| 1. Attendees are prevented from accessing resources they are not authorized to access, even by manipulating the URL. |
| 1. When an attendee account is deleted, all reviews written by that attendee are also deleted. |
| 1. When an event is deleted, all associated reviews for that event are also deleted. |
| 1. Both the price and number of seats left are accurately calculated and displayed. |
| 1. All components have the attributes from the requirements document. |

| **Extra credit** |
| --- |
| 1. This project correctly implemented a search function for the admin to use. The input is the event name; the search result lists attendees who booked this event. |
| 1. This project implemented a function to allow an attendee to buy a ticket for another attendee (the ticket can be viewed by both the attendee who pays for the ticket and the attendee who receives the ticket). |

| **Workflow** |
| --- |
| 1. The workflow is intuitive. Would you suggest making any changes to the workflow? |
| 1. Overall, do you find other problem(s)? Please specify them. |

# Round 2 Rubric

| **Software Engineering and Testing** |
| --- |
| 1. The code is written in a clean and readable way. |
| 1. Each method performs only one task. (One method should only handle one task, if there are multiple tasks, there should be function calls. Mention any relevant details.) |
| 1. Variable and method names are indicative of what the variables are storing/handling. (Mention any relevant details.) |
| 1. Commit messages are indicative of what changes were made in the commit. (Mention any relevant details.) |
| 1. This system works as it is supposed to. (If you found any problems in the first round, did the authors fix them? Comment on any functionality that is still failing.) |
| 1. This team made commits in round 2. |
| 1. The README file contains all the information needed, and the code is well documented, with adequate comments to explain the coding. |
| 1. Has the testing been done properly for at least one model? |
| 1. Has the testing been done properly for at least one controller? |

| **Design and Workflow** |
| --- |
| 1. Overall, do you find other problem(s)? How would you suggest fixing it/them? |