Assignment-Ruby-on-Rails-Book-Collection 2

Objective

The primary objective of this assignment is to help you get familiar with TDD testing using the simple CRUD Application you built previously.

Task description

Taking the initial Book Collection Application you developed, implement Test-driven development (TDD) using RSPEC by writing unit and integration tests, after which you will resolve bugs.

Submission

1. Deadline: See Canvas
2. You will use your own Github for development, however, you will push your source code to the GitHub Classroom repository at Github Classroom
3. Submit a screenshot of an analysis of your code (e.g., testing against 1 sunny day and 1 rainy day) against the Canvas rubric using the LLM of your choice. This should communicate improvement suggestions to comply with the given rubric.
4. Submit the report (described below) to Canvas.

Assignment Instructions

1. Write unit/integration tests for the code you wrote for the "Book Collection 1" assignment

    - Write a unit test using RSPEC to test the model for 'book' and an integration test to make sure that the book is added successfully to the database as indicated by the correct 'flash notice.' Likewise, test a 'rainy-day' scenario by trying to save a book with a 'title' blank, where you look for the corresponding 'flash notice.' The tests should pass. **Take a screenshot (1) of the results. You'll need to put the screenshot into your report later.**

    - Next, you need to write more tests on your own for new attributes – author (String), price (Number), and published date (Date). For each new attribute, you need to write a unit test and integration test. After you write all the tests, run these tests. Your tests should **not** pass at this moment because you haven't started coding for these other attributes yet. **Take a screenshot (2) of the results. You'll need to put the screenshot into your report later.**

    - Insert code for the additional attributes author (String), price (Numerical), and published-date (Date/DateTime). **Take a screenshot (3) of the results of the tests passing after the successful inclusion of the attributes. You'll need to put the screenshot into your report later.**

      Note: published-date should be a drop-down menu.

    - Remember to always make the migrations to create a database table

    - Make sure you follow Rails convention and architecture (MVC architectural design pattern).

    - After completing all of the requirements, run all the tests again, they should all pass. If not, refactor your code until they all pass. When all of them pass, **take a screenshot (4) of the results. You'll need to put the screenshot into your report later.**

2. Create a new branch test

   $ git checkout -b test

3. Save all code

   $ git add .

   $ git rm node_modules (if node_modules is in your .gitignore, this step is not necessary)

   $ git commit -m "your commit message"

4. Push the changes to your GitHub repository. To check the remote, use $ git status.

   $ git push origin test

5. Add a screenshot of your pull request to merge to the main branch - screenshot (5)
6. Add a screenshot of your commit history in your branch - screenshot (6)

Report

Your report should include the following content:

1. Link to your Github repository where you committed the working code.
2. Screenshots. Please add them sequentially from (1) to (n). For each screenshot, explain what it is about.

Save your report as FirstName_LastName_Lab-BookCollection 2-TestDrivenDev(TDD).pdf and submit it to your teams individual submissions folder .

Grading

Will be graded using the following sections of the Sprint Rubric:

- **Quality - Correctness**

References: Linkedin Trainings free of charge with your TAMU credentials

Ruby on Rails 7 Essential Training

Article:

Ruby On Rails - HTTP, MVC and Routes