Thomas Orth

Assignment 5 - Open Source Software: Project Implementation

Full path to repo for project: https://www.github.com/TomOrth/Open-Learning-Platform
VM Name: csc415-server34.hpc.tcnj.edu
VM Account: student1
Full VM Path to Project folder: /home/student1/Open-Learning-Platform
Full VM Path to src folder (contains the Rails application):
/home/student1/Open-Learning-Platform/src
URL of the application once the VM is running: http://csc415-server34.hpc.tcnj.edu:3000/

NOTE WHEN STARTING THE SERVER:

Amid the start up of the server, this warning stack will appear:

/home/student1/.rbenv/versions/2.7.0/lib/ruby/gems/2.7.0/gems/actionpack-6.0.2.1/lib/action_dispatch/middleware/stack.rb:37: warning: Using the last argument as keyword parameters is deprecated; maybe ** should be added to the call
/home/student1/.rbenv/versions/2.7.0/lib/ruby/gems/2.7.0/gems/actionpack-6.0.2.1/lib/action_dispatch/middleware/static.rb:110: warning: The called method `initialize' is defined here
/home/student1/.rbenv/versions/2.7.0/lib/ruby/gems/2.7.0/gems/activerecord-6.0.2.1/lib/active_record/type.rb:27: warning: Using the last argument as keyword parameters is deprecated; maybe ** should be added to the call
/home/student1/.rbenv/versions/2.7.0/lib/ruby/gems/2.7.0/gems/activerecord-6.0.2.1/lib/active_record/type/adapter_specific_registry.rb:9: warning: The called method `add_modifier' is defined here

These warnings are NOT generated by my code. This is due to an internal issue within rails due to the new version of ruby deprecating somethings that the libraries that rails uses. The server otherwise compiles and runs without errors or warnings caused by my code.

Summary of functionality implemented

All use cases that were proposed to be completed this semester have been completed

1. Registration and Login for Educators
2. Upload verification paperwork for Educators
3. Admin views verification paperwork and can verify educators
4. Educator can create a lesson plan
    a. Able to upload files
    b. Able to select a topic for the lesson plan to belong to

    c. Able to update lesson plans
    d. Able to delete lesson plans
5. Educators can search for lesson plans on the home page
    a. Search bar searches for matching string in the title and the dropdown next to the search bar allows for filtering by topic

Summary of functionality not implemented

All use cases that were marked as done if time permits and implemented at another time. This is because there was not enough time to work on these use cases this semester.

Known Bugs/Limitation

- No known bugs
- Limitations
    - The storage blobs for ActiveAttachment (The library for activerecord to store files) are not deleted when Lesson Plans are deleted
        - This is posted to the Github as an Issue to be fixed
    - No restriction on the files that a lesson plan can have
        - Research and polling of K-12 teachers would be needed to determine which file types they would need

Installation/Usage Guide

This is the instructions to install and use the application. For VM specific instructions, please go to the “Running on the VM” section of the document. Otherwise, go to the “General Installation” section.

Running on the VM

1. Run the command: ssh student1@csc415-server34.hpc.tcnj.edu
    a. This requires the VPN to be on to use
2. When prompted, input the password: Eng1neer1
3. Once you are on the VM, run: cd Open-Learning-Platform/src
4. Run the following command to start the server: rails s --binding=10.18.6.34
5. Once this is performed, go into your web browser (Please use chrome. The web application has only been tested in Chrome so I do not know if the rendering engine for Firefox would work with the front end assets) and enter this url: http://csc415-server34.hpc.tcnj.edu:3000/
6. You will now be able to use Open Learning Platform

General Installation

This assumes that you have installed Ruby 2.7.0, Rails 6.0.2.1, Postgres, NodeJS and Yarn
This application has been tested on a Linux environment so currently is the only known one that the server can be run on.
For ruby, rails and postgres, please follow this: https://gorails.com/setup/
For nodejs, follow the appropriate guide for your OS:
https://nodejs.org/en/download/package-manager/
For yarn: https://classic.yarnpkg.com/en/docs/install/#mac-stable

1. Clone the repository by performing: git clone https://www.github.com/TomOrth/Open-Learning-Platform
2. cd Open-Learning-Platform/src
3. Run: bundle install and yarn install
    a. If bundler is not installed, please run: gem install bundler -v 2.1.4, and then repeat the above
4. Run the following commands to properly setup the database
    a. rake db:create
    b. rake db:migrate:reset
    c. rake db:migrate
    d. rake db:seed
5. Run the command: rails s
6. Now go to Google Chrome (Browser this application has been tested in), and go to localhost:3000
7. You are now able to access the application

Research about how projects maintain open source projects

This is the research for the guidelines of maintaining this project. To read the guidelines instead, please skip to the “Open Source Maintenance and Contributions” section.

Major projects examined:

https://github.com/freeCodeCamp/chapter
https://github.com/tensorflow/tensorflow
https://github.com/apache/airflow/
https://github.com/grafana/grafana

These projects listed above share some common patterns. They all discuss issues in a third party service that is not Github. For chapter, it’s discord. For airflow and grafana, it’s slack. For tensorflow, there is a google group discussion board.

Each project has a separate contribution guideline to follow for contributing to the codebase and criteria of pull request acceptance. For chapter, they require that developers read their code of conduct and join their discord server in order to be in the loop and make sure developers are working on open issues not currently being handled by other developers. Once the developer has an issue identified, they fork the repository and create a new branch from master for the issue they are working on. The branch is prefixed with fix/, feat/, and docs/. Once complete, the dev makes a pull request (PR) and has required information in their PR such as a dev checklist that chapter asks from each developer and a summary of the PR. For tensorflow, the developer is just asked to find an issue on the issues tab and start working. The main thing the maintainers ask is that the developers create and update any relevant unit tests. Note though they also are required to read a code of conduct. Similarly, Airflow requires this from their contributors, also with an emphasis on including tests and coverage reports. Finally, grafana also has a code of conduct to review before developing but does not have a focus on tests.

There are some commonalities between the projects. For one thing, majority of the projects require that all contributors fill out what is called a Contributor License Agreement (CLA) which wikipedia (https://en.wikipedia.org/wiki/Contributor_License_Agreement) defines as a way to define contributions to intellectual property. They all have some form of Continuous Integration such as Github Actions or CircleCI. In tandem with this, in order for PRs to be approved and merged, the CI builds need to pass and developers will then inspect the code of the PR to make sure it is following good practices. Finally, each project utilizes the Github Projects feature for their project, which gives a Trello-like board interface for project management.

Open Source Maintenance and Contributions

Based on the research done in the above section, the guidelines for maintaining and contributing to the project are defined as below.

Discussion of Issues/Bugs and Project Future

For discussions of issues/bugs as well as the future of the project, a slack workspace would be used. Developers would discuss open issues on the project and the best way to maintain the project. This would include the future of the project in order to make sure it is well maintained.

Issue Creation

Issues that are created will be marked according to the type of contribution it is: bug, feature, or documentation. A bug is an issue that is related to a bug with the current application. Feature is an issue with proposed functionality that should be added. Documentation would be an issue about the addition and maintenance of the project documentation.

Contributing to the project

Developers will follow these guidelines below. Note the workflow is adopted from the workflow for projects called git flow (https://nvie.com/posts/a-successful-git-branching-model/). I've elected to follow this workflow as it is similar to what chapter follows but is more in-depth for things such as hotfixes and uses a separate dev branch before going into master:

1. Fork the repository (if not already done)
2. Branch off of the develop branch and name a branch for the action you are doing
3. If it is added functionality, the branch will be called feature-<short desc of purpose>. If it is a bug fix, the branch will be named fix-<short desc of purpose>.
4. Add the code needed.
5. Submit a PR (see the section labeled "Pull Request Review Process" for this document).

There are two other types of branches that could be made. These typically would be handled by those who are identified as head maintainers of the project.

The first is a hotfix branch. If there is a bug in the current production release of the codebase, which is on the master branch, a quick fix is made in order to ensure that there is a clean working version of the project on master.

Steps:

1. Fork the repo (if not already done)
2. Branch from master and label the branch hotfix-<short desc of purpose>
3. Once the fix is made and tested successfully, submit a PR to master and develop. Due to this being a hotfix, it would need to bypass the review process and be cleaned up afterwards.
4. Merge both PRs into their respective target branch.
5. Delete the hotfix branch.

The other branch type is release branch. This is to prepare for a new release of a production build of the application

Steps to create a release branch

1. Branch off of develop
2. Perform last minute maintenance of any kind before release
3. Merge back into develop and also into master via a PR (follow formal review process in the "Pull Request Review Process" section)
4. Prepare a release tag for github to benchmark the release of the project
    a. Follow semantic versioning - https://semver.org/

Pull Request Review Process

When developers create a PR, they must have a descriptive title of what their PR is doing. In the summary/description of the PR, the developer must properly summarize how their functionality works. They must also reference the related issue properly by typing “Resolves #<number>”. By doing this, Github will automatically close the issue if the PR is merged. In the contents of the PR, unit tests will be updated or created as needed. Additionally, steps to test the PR must be included so that the reviewer can see for themselves that the project behaves as required. Test coverage reports should be included in the PR for maintenance of the project. In addition to this, the developer must adhere to these requirements:

1. Add or Update the maintenance information in the code files
2. Follow the YARD tag style of code documentation (https://www.rubydoc.info/gems/yard/file/docs/GettingStarted.md)
3. Adhere to the code of conduct (will be created when there is an established developer community)

A reviewer will be assigned to review the PR. The following review process will be followed by any reviewers:

1. Ensure that the PR has a passing build for continuous integration. If this is not the case, the PR will not be reviewed until then as the tests are not passing and the PR is not stable.
2. Each modification will be reviewed to ensure that the above mentioned criteria for developers is followed and the coding style set by the developer team is followed.
3. Any infraction ID'd by the reviewer must be fixed.
4. Once the PR is satisfactory and the reviewer manually tests the PR, the PR will be approved, then merged into the target branch
5. Once all PRs for that branch are completed, the branch will be deleted.

Managing Issues

Outside of the community slack, Github Projects will be used. There will be three columns in the main board: Todo, In progress and finished. This will be used to maintain issues and be a reference to the development team.

Continuous Integration

The project will use Continuous integration. The service that will facilitate this is TravisCI (https://travis-ci.org/). Continuous Integration is used to ensure that the state of the project is always known and failing tests are made clear.

Unit tests and Integration tests

Unit tests would be written in order to ensure that functional requirements and security requirements of the project are met. This can be, and will be, created with minitest (https://github.com/seattlerb/minitest).

Regression Testing

Since continuous integration will run for all PRs and commits, regression testing will happen simultaneously with Unit and Integration as if new functionality breaks old functionality, then the tests will fail.

Test Case Document (Revised from Assignment 3):

Note: After testing in class, the database will be cleared in order to allow for the same credentials to be used for testing later on

For any use case that it is noted to be logged in as one user but you are a different type i.e. Test case says to ensure you are an educator but you are logged in as admin or vice versa, log out of that user account by clicking “Log-Out” on the navbar

Obviously, if you are logged in as the correct user i.e. the test case asks for an educator and you are an educator, stay logged in

Functionality Tested	**Inputs**	**Expected Output**	**Actual Output**
Educator Registration	Click “Educator Log-in” -> Click “Sign up” Email: test@test.com Password: testing123 First name: John Last Name: Doe Click “Sign Up”	Success message displayed, redirection to homepage	
Admin Registration (ensure you are logged out by clicking logout on the navbar)	Click “Admin Log-in” -> Click “Sign up” Email: admin@admin.com Password: testing456 Click Sign up	Success message displayed, redirection to homepage	
Educator Login (be sure to be logged out)	Ensure you are logged out (Click	Success message displayed, redirection	

	Logout on navbar) Home page -> Educator Log-in Email: test@test.com Password: testing123 Click “Log in”	to home page	
Admin Login (be sure to be logged out)	Ensure you are logged out (Click Logout on navbar) Home page -> Admin Log-in Email: admin@admin.com Password: testing456 Click “Log in”	Success message displayed, redirection to home page	
Invalid credentials for login of educator	Ensure you are logged out (Click Logout on navbar) Home page -> Educator Log-in Email: test@test.com Password: wrong123 Click “Log in”	Error message, saying invalid email or password	
Upload verification paperwork for educator For this use case, download the verification file by clicking here (right click the link and open in another tab) Ensure that you are logged in as an educator. If not, follow the Educator Login test case (If logged in as Admin, click Logout on the navbar)	Home page -> Profile -> Click Verification tab -> Click choose file -> Select verification document noted in the left column A PDF file	Success message that paperwork was submitted	

Prevent educator actions until verified Ensure that you are logged in as an educator. If not, follow the Educator Login test case	Click on “Plans” tab on educator profile	A display should be shown, saying that they cannot do anything until the educator is verified	
Admin verifies Educator Ensure that you are logged in as an admin. If not, click logout on navbar and then follow the Admin Login test case	After logging in as Admin, Click Profile -> “Verification” tab of the admin profile, click verify next to the educator with the name “John Doe”	A success message that the educator is now verified should be shown	
Educator can now do actions Ensure that you are logged in as an educator. If not, follow the Educator Login test case	Logout of Admin by clicking Logout. Follow login for educator via the Educator Login Test Case. Click profile. Under the educator profile, go to the “plans” tab	There should now be a “New lesson plan” button	
Educator creates a lesson plan Download PDF 1 here Download PDF 2 here Ensure that you are logged in as an educator. If not, follow the Educator	Click the “New lesson plan button” Title: Test Plan Description: I am meant to help educate Choose “Science” for the topic Content (Click choose file): The two pdfs that were just downloaded. You can choose multiple files from the file explorer	A success message displayed, redirect to profile, a new lesson plan should be displayed with the name “Test Plan”	

Login test case	prompted		
Educator can view lesson plan documents	On the plans tab, Click show next to the lesson plan. Under contents, click on either file. It will open the PDFs in another tab	PDFs open in another tab	
Educator updates a lesson plan	Click on “Edit” next to the lesson plan. Change the title to “New Test Plan”, the description to “I am now a different plan” and two different files	A success message should be displayed, redirection to profile, and now there should be a lesson plan in the list “New Test Plan”	
Educator searches for lesson plans	Home page Input for search: “New”, topic: “All”, click submit/search	On that page should be a list containing the lesson plan “New Test Plan”	
Educator search returns no lesson plans	Home page Input for search: “Z” Click submit/search	On that page, a message in the blank area should say “No lesson plans found”	
Educator deletes a lesson plan	Click Profile -> Plans tab -> Click “Destroy” next to the lesson plan “New Test Plan”	Success message displayed, the lesson plan should no longer show up	