Flask based Web App Project

Group Project

Students will work in groups of 3 to 4 to create and deploy a Flask Web Project consisting of three co-operating pieces. The source code must be tracked in a private GitLab repo shared with the instructor (ddubois1), where group members use the feature branch (protected branch) development style. The project must meet the requirements outlined below.

Contents

Learning Objectives 1
Project Overview 2
Project Requirements 2
User Management Application 2
Website Administration Application 2
Course and Competency Display and Management 3
Development Practices and Process 3
Test Data 4
Development Tools 4
Progress Reports 4
Deliverable (due April 28, 2023) 5

Learning Objectives

- Evaluate requirements and translate them into a web application design
- Create a server-side web application
- Access a database to create data models
- Use templating and server-side rendering to show models
- Design REST APIs to interact with other automation tools
- Deploy a web application to a server
- Communicating project status and progress
- Work in teams using the feature branch (protected branch) model to implement an application

Project Overview

The website will serve as a system to interact with the course list service database used to represent the computer science programs and its competencies. Leveraging the provided database, the website will provide a user interface to allow users to display information related to courses, competencies, and their interconnection. Additionally, the website will provide a user account and group system allowing only authorized users to make changes or view certain data. Finally, an admin dashboard will be provided to manage users, their roles, and account details. The term website is used to refer to the entire project from this point on in the specification.

Project Requirements

User Management Application

The website must provide a login and access control feature, where the website is made up of two sets of users:

1. Visitors who are not registered and do not log in
    a. They can access and browse some of the websites content
    b. They can choose to register and create an account
2. Members are registered users who have an account
    a. They can login and logout
    b. Reset their password
    c. They have a profile associated with their account that contains the following information
        i. Name
        ii. Email address
        iii. Avatar (picture representing them)

Note, through the rest of the document Visitors and Members are used to describe the behaviours above. Users is used as a catch all term to refer to anyone interacting with the website. See the Website Administration Application for more information.

Website Administration Application

The website must have an administrative dashboard where the registered users (members) can be manipulated and groups can be created. This will be used to provide access control to features in the other applications. Depending on the groups, users should be able to see what members are a part of each group. The site must support the following groups:

- Members
    - Users of this group have the privileges of Members as described in the spec. This group represents the registered members. By default, new members are placed in this group. Users in this group can only see a list of other members and cannot see their group.
- Admin_user_gp
    - Users of this group can add, delete, block, and manage users in the Members group. This represents user administrators
- Admin_gp

  - Users of this group have all the privileges of the group above. In addition, they can add, move, and remove users from the various groups.
  - Users in this group can see the groups of all other users.
  - A special user account called “Instructor” with password “Python420” must be created and placed in this group by default.

Course and Competency Display and Management

The application must provide a system for Members to create, delete, and modify, courses, competencies, and their connections.

Members and visitors (those who are not registered and logged in) must be able to browse, filter, and search through existing courses, competencies, and their connections. This is the primary view of the website and makes up the home page. However, only Members can edit a course, competency, or connection.

A user-friendly view of the courses, competencies, and their intersection must be provided for a given program. Additionally, a validation section should be present to report an existing error with the competencies and courses for a given program.

Retrieving information about a course, competency, or their intersection be provided through a REST API. Users should be able to query information about a course, competency, or their relationship using a tool such as Curl interacting with the REST API.

Development Practices and Process

The following development practices must be applied:

- Proper class design must be followed including
  - Separation of concerns
  - Loosely coupled functionality
  - Inheritance
  - Exception handling
  - Following MVC
- REST APIs must be used for some of the functionality
- Proper naming convention (pep-8)

Group members are expected to follow the following development process:

- Use Git and Gitlab to manage code
- Using the feature (protected) branch model when developing features
- Using Merge Requests to integrate feature (protected) branches
- Ensuring commit messages are meaningful and valuable
- Performing code reviews as part of the Merge Request

Note, individual marks will be associated with each group member. Their contribution to the project will be measured using Git and Gitlab, so be sure to commit and work on code with your associated user. Code reviews are considered part of the individual contribution.

Note, students who do not pass the individual portion of the project will not pass the project overall and can obtain a maximum grade of 50%.

Test Data

To verify the functionality of the website, the following data and models must be present in your final submission:

- 10 Member users
- 2 Users in each admin group
    - Ensure you have the special "Instructor" user as described above
- Populated data from provided database

Development Tools

The following tools and frameworks must be used when developing the web application:

- HTML, CSS, JavaScript for client-side web content
- Flask as the server-side web framework
- Git as the version control system
- PDBORA19C as the server-side database

Progress Reports

At the end of each week, a report must be submitted that outline the progress of the project. These reports will be graded and contribute to the overall grade of the project. A table is provided as a sample of how the report can look. It must include the following information:

- URLs to the merge requests representing the completed features, or on-going features of the week.
- A summary of the overall status of the project
    - What tasks have been completed (with links to Merge Requests)
    - What tasks are schedule for the following weeks (what will become Merge Requests)
    - Brief estimates of each task

ID	Task	M.R. URL	Assignee	Start Date	Duration	Problems	Status	Note
T1	Item Catalog Models	LINK…	John	April 5	2 hours	N/A	100%	No data in the DB yet
T2	Admin Models	LINK…	Jane	April 6	4 hours	Difficulty finding data types	50%	

Deliverable (due April 28, 2023)

Submit a zip called Project2 through Moodle of the Gitlab repo. Additionally, the application needs to be deployed and available on your server, where features can be tested. Ensure that your code runs and has no syntax errors. Ensure your code follows Pep8 standards and is appropriately commented.

The following files must be included along with your source code

- Readme.md
  - Include a link to your Gitlab repo
  - Ensure all development setup steps are well described
  - Ensure all deployment steps are well described
  - Ensure the names and student ids of group members are listed
  - A link to the location where the demo application is deployed.