Databases Assignment

Building a Web Application with Flask and a Database

How-To Guide

The Databases Assignment represents 12% of the overall course grade. Marks will be awarded out of 100% under the following marking scheme.

Instructions

Integrate PostgreSQL Database with Flask: To take your web development to the next level, you will integrate a PostgreSQL database with Flask. This integration will allow you to store and manage data related to your web application efficiently. You can utilize the database to store project information, user messages, contact details, or any other relevant data.
You will also incorporate modern HTML, CSS, and JavaScript to create an engaging and visually appealing user experience. The web application can be tailored to your interests and needs, providing you the flexibility to showcase your projects, skills, or any other content you desire.

Project Details

A. **Set up the Flask Environment:**
   - **a.** Install Flask by running the command: **pip install flask**.
   - **b.** Create a new directory for your Flask project.
   - **c.** Initialize a virtual environment by running the command: **python -m venv venv**.
   - **d.** Activate the virtual environment:
     - i. For Windows: **venv\Scripts\activate**
     - ii. For macOS/Linux: **source venv/bin/activate**

B. **Create Flask Application Files:**
   - **a.** Create a new Python file named **app.py** and import the necessary Flask modules.
   - **b.** Set up the Flask application object and define routes for different sections of the website, such as home, projects, skills, about, and contact.

C. **Create HTML Files:**

a. Create at least five HTML files.
b. You should have a base template with the head and any other fixed elements like the nav, and then have all other html files can inherit from the base template and have the block(s) that are unique to them.
c. Design the structure and layout of each HTML file to create a cohesive and visually appealing website.

D. Apply CSS Styling:

a. Create a new directory named **static** in your project's root directory.
b. Inside the **static** directory, create a subdirectory named **css**.
c. In the **css** directory, create a new CSS file named **style.css**.
d. Write modern CSS code in the **style.css** file to style your HTML pages. Use selectors, classes, and IDs to target specific elements.
e. Apply appropriate styles to create an attractive and professional website.

E. **Implement JavaScript Functionality:**

a. Create another subdirectory inside the **static** directory named **js**.
b. In the **js** directory, create a new JavaScript file named **script.js**.
c. Write JavaScript code in the **script.js** file to add interactivity and dynamic behaviour to your portfolio website. Use features like smooth scrolling, image sliders, or any other relevant functionality.

F. Integrate HTML, CSS, and JavaScript with Flask:

a. Link the CSS file to each HTML file using the **<link>** tag in the head section.
b. Link the JavaScript file to each HTML file using the **<script>** tag before the closing body tag.

G. Define Flask Routes and Render HTML Files:

a. In the **app.py** file, define routes for each section of the website.
b. Use the **render_template** function from Flask to render the respective HTML files for each route.
c. Implement SQLAlchemy model classes as well as any other custom classes and data structures appropriate to manage data within the Flask application.
d. Implement Flask routes that handle different HTTP methods (GET, POST, etc.) and perform appropriate actions based on the user's requests.
e. Feel free to explore additional Flask extensions or libraries to enhance your project.

H. **8. Integrate PostgreSQL Database with Flask:**

a. Install Flask-SQLAlchemy to facilitate database integration.
b. Configure the Flask app to connect to the PostgreSQL database.
c. Integrate the model classes with the database connection.
d. Implement necessary Flask routes to handle database CRUD operations.

I. **Test and Run the Flask Application:**
   a. Save all your files and ensure that the Flask application is running.
   b. Open a web browser and enter the URL provided by Flask to access your portfolio website.
   c. Test each route and verify that the respective sections and pages are displayed correctly. You can use automated tests here.

J. **Hosting the Web App on Render.com:**
   a. Connect your project to your GitHub repository containing the Flask app.
   b. Create a web service on Render.
   c. Configure the PostgreSQL database.
   d. Configure the necessary build settings and environment variables on Render.
   e. Deploy the Flask app to Render by triggering a build.
   f. Wait for the build and deployment process to complete.
   g. Access your hosted web app on the provided URL from Render.

Preparing Your Submission

- Compress all your project files into a single ZIP file, including the Flask app files and the deployment configuration files.
- Submit the ZIP file along with clear instructions on how to deploy and access the web app on Render.com.
- Ensure that your code is properly structured and follows best practices.
- Implement additional features or design enhancements to showcase your creativity and skills.
- Use comments including docstrings to explain each non-trivial function, and a README file for an overview.
- Make sure your Flask app is compatible with Render's requirements and configurations.
- Ensure that your web app is accessible and fully functional on the provided Render URL.
- Double-check the deployment instructions and any necessary configuration details to ensure a smooth deployment process.
- Include any additional details or considerations specific to the deployment on Render.com in your submission.
- Verify the hosted web app's functionality and appearance after deployment to ensure it matches your local development version.

How You Will Be Assessed

The following rubric describes how the essay will be assessed:

1. Demonstrate an understanding and apply the key concepts of a database schema and crud logic.	**2. Show clear understanding of Database integration in Flask.**	**3. Show clear understanding of integrating CSS styling and JavaScript functionality in line with modern design.**	**4. Demonstrate clean code structure, organization, and adherence to best practices.**	**5. Show evidence of a hosted web app that is fully functional and accessible**
Distinction Criteria				
Demonstrates strong ability to apply the key concepts of a database schema and crud logic.	Demonstrates strong ability to apply key concepts and principles of Database integration in Flask.	Demonstrates a strong understanding of integrating CSS styling and JavaScript functionality in line with modern design.	Demonstrates a strong ability to create a clean code structure, organization, and adherence to best practices.	Demonstrates strong evidence of a hosted web app that is fully functional and accessible
Merit Criteria				
Able to apply the key concepts of a database schema and crud logic.	Able to apply key concepts and principles of Database integration in Flask.	Demonstrates a good understanding of integrating CSS styling and JavaScript functionality in line with modern design.	A very good level of ability to create a clean code structure, organization, and adherence to best practices.	Demonstrates a good evidence of a hosted web app that is fully functional and accessible
Pass Criteria				
Adequate use of the key concepts of a database schema and crud logic.	Adequate use of the key concepts of Database integration in Flask.	Makes adequate use of integrating CSS styling and JavaScript functionality in line with modern design.	Adequate evidence for creating a clean code structure, organization, and adherence to best practices.	Adequate evidence of a hosted web app that is fully functional and accessible
Unsatisfactory Criteria				
Response is partial or tangential. Requires greater depth, level of detail and discussion.	Response is partial or tangential. Requires greater depth, level of detail and discussion.	Inadequate application of course learning integrating CSS styling and JavaScript functionality in line with modern design.	Inadequate use for creating a clean code structure, organization, and adherence to best practices.	Unsatisfactory evidence of a hosted web app that is fully functional and accessible
Clear Fail Criteria				
Little evidence of a database schema and crud logic.	Little evidence of Database integration in Flask.	Little evidence of ability to apply key course concepts in practice.	Very little evidence of for creating a clean code structure, organization, and adherence to best practices.	Little evidence of a hosted web app that is fully functional and accessible
No Attempt Criteria				
No submission	No submission	No submission	No submission	No submission