**Prog3660s**

**Community Hub Web Application**

**Objective:** Develop a web application named "Community Hub" using Python, Flask, and MongoDB. This application will be a platform where users can share and view community events, posts, and discussions. The project must demonstrate your ability to perform all major database operations (Create, Read, Update, Delete - CRUD).

**Requirements:**

1. **Backend Development:**
   * Utilize Flask as your web framework.
   * Use Python for all backend logic.
   * Connect to a MongoDB database for data storage.
2. **Database Operations:**
   * **Create:** Users should be able to **securely** create an account and add new posts or events.
   * **Read:** Implement functionality to view all posts and events, and user profiles.
   * **Update:** Users must be able to update their profiles and edit their posts and events.
   * **Delete:** Allow users to delete their posts, events, and account.
3. **Frontend Development:**
   * Create a user-friendly interface.
   * Ensure the application is responsive and works on different devices.
   * Use HTML, CSS, and JavaScript (only as needed) for frontend development.
4. **User Authentication:**
   * Implement user registration and login functionality.
   * Ensure secure handling of user credentials.
5. **Functionality:**
   * Include features such as commenting on posts, liking events, and searching for specific posts or events.
   * Implement sorting and filtering options for posts and events.
6. **Documentation:**
   * Provide a README file explaining the functionality of the application, how to set it up, and how to connect to the database.
   * Comment your code adequately for clarity.
7. **Testing:**
   * Write tests for your application to ensure all functionalities work as expected.
   * See [here](https://machinelearningmastery.com/a-gentle-introduction-to-unit-testing-in-python/) for a guide on writing unit tests for core program functionality.
8. **Reliability and Scalability:** 
   * This might include implementing advanced monitoring solutions, creating efficient logging systems, or setting up automated backup and recovery processes.
9. **Replicability:**
   * Be sure to use ‘venv’ and a ‘requirements.txt’ file for your Python environment.
   * Optionally, include a Dockerfile to build the environment (including Mongo).
10. **Deployment:**
    * Optionally, deploy your application on a platform of your choice (like Replit or AWS) for live interaction.

**Deliverables:**

Submit your project in a GitHub repository containing all the code, a README file, and any other necessary documentation. Also, provide a link to the live version of your application if deployed.

**Evaluation Criteria:**

* Functionality and completeness of all CRUD operations.
* Code quality and organization.
* User interface design and responsiveness.
* Documentation and ease of setup.
* Reliability assurance and unit testing to validate functionality.
* Bonus: Environment setup, deployment, and live interaction.