RUDI JUANDI BUTARBUTAR

is passionate about data exploration, with expertise in backend development, database management, and software development. Developed strong skills in C, Python, Golang and SQL to deliver efficient and scalable solutions.

Project Name

Enterprise - 2019

Project Name

Enterprise - 2019

Project Name

Enterprise - 2019

Project Name

Enterprise - 2019

Project Name

Enterprise - 2019

Project Name

Enterprise - 2019

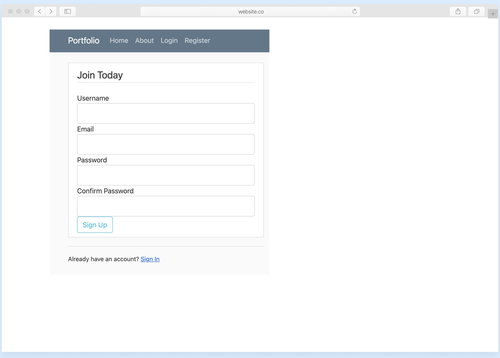
Backend Development
User Function with Flask

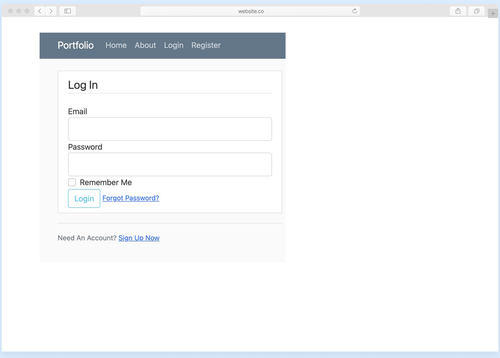
developed a user authentication and management system using Flask as the web framework, Python for the backend logic, and SQLAlchemy for database interactions.

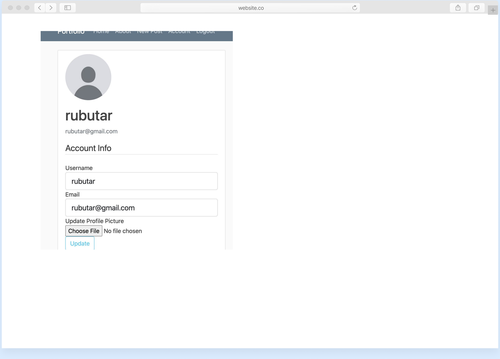
The core functionality includes:

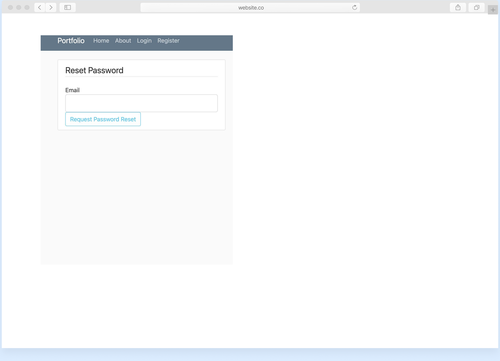
1. User Registration: New users can create accounts by providing essential details like username, email, and password. The password is securely hashed using libraries such as werkzeug.security before being stored in the database.
2. User Login: The login function verifies user credentials against the stored hashed password. On successful authentication, a session is created to maintain the user's logged-in state.
3. User Profile Management: Authenticated users can view and update their profile information. SQLAlchemy's ORM capabilities simplify the querying and updating of user data, ensuring seamless interactions with the database.
4. Database Integration with SQLAlchemy: SQLAlchemy is used to define the User model and manage database operations. The ORM approach facilitates easy mapping between Python objects and database tables, allowing for complex queries and efficient data manipulation.
5. Forgot Password with Token-Based Security: The "Forgot Password" functionality allows users to reset their passwords securely. When a user requests a password reset, a unique token is generated and sent to their registered email address. This token is securely stored and linked to the user in the database using SQLAlchemy.
   - Token Generation and Storage: The token is generated using a secure random function and is time-limited to prevent misuse. SQLAlchemy manages the storage and association of this token with the user in the database.
   - Password Reset: When the user clicks the link in their email, the system verifies the token's validity. If valid, the user is prompted to enter a new password, which is then hashed and updated in the database, replacing the old password.

Screens









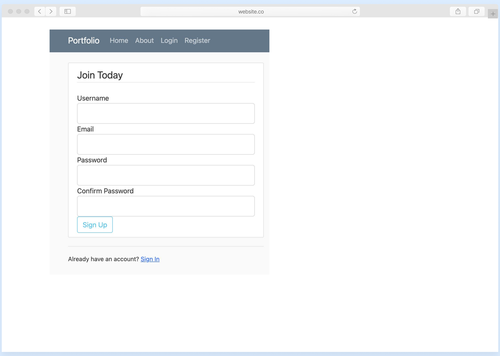
Backend Development
Post Function with Flask

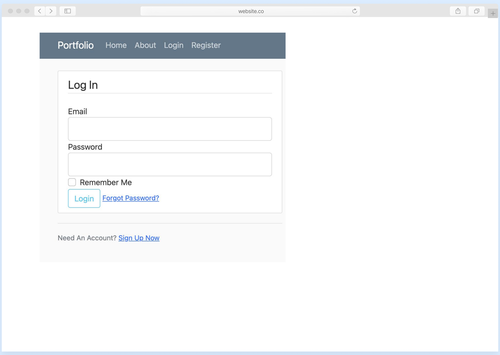
developed a robust post creation and management system using Flask as the web framework, Python for backend logic, and SQLAlchemy for database management. This system forms the core of a content-driven application where users can create, view, edit, and delete posts.

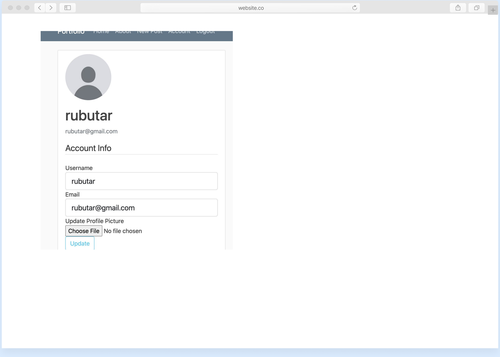
Key functionalities include:

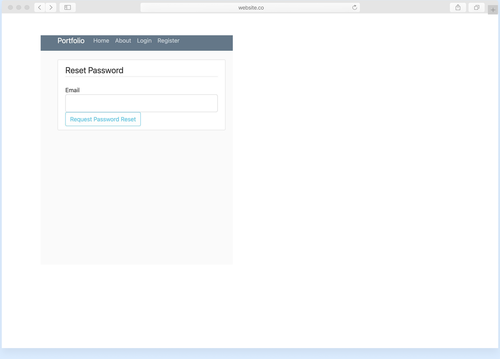
1. Post Creation: Authenticated users can create posts by submitting a form that captures essential details such as title, content, and optional tags. Flask handles the form data, which is then processed and stored in the database using SQLAlchemy.
2. Database Integration with SQLAlchemy: The Post model is defined using SQLAlchemy's ORM, allowing seamless mapping between the database and Python objects. SQLAlchemy manages all database operations, ensuring that data is efficiently stored and retrieved.
3. Post Retrieval and Display: The application retrieves posts from the database and displays them to users. This includes features such as pagination, filtering by tags, and sorting by date or popularity, all handled by SQLAlchemy queries.
4. Post Editing: Users can edit their posts after creation. The edit functionality retrieves the post from the database, pre-populates the form with existing data, and updates the post upon submission. SQLAlchemy ensures that the changes are correctly applied and saved in the database.
5. Post Deletion: Users can delete their posts, which removes the post from the database. Flask handles the deletion request, and SQLAlchemy processes the database operation, ensuring that the associated data is cleanly removed.
6. User Authentication and Authorization: To manage posts, users must be authenticated. Role-based permissions are enforced to ensure that only authorized users can edit or delete posts. Flask's session management and SQLAlchemy's ORM make it easy to associate posts with specific users and enforce these permissions.

Screens









Project Name

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse molestie leo sem, id porttitor nunc ultricies dignissim. Nullam consectetur dictum nibh, sed volutpat nunc aliquet id. Curabitur libero diam, egestas in volutpat id, scelerisque et ipsum.

Before After

Screens

Project Name

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse molestie leo sem, id porttitor nunc ultricies dignissim. Nullam consectetur dictum nibh, sed volutpat nunc aliquet id. Curabitur libero diam, egestas in volutpat id, scelerisque et ipsum.

Before After

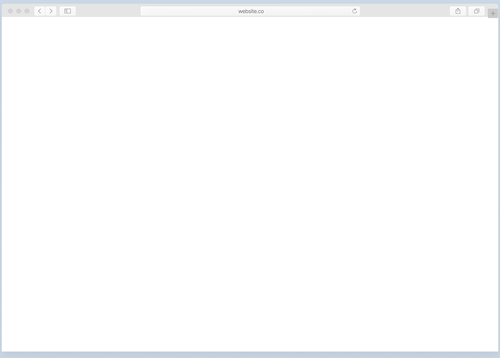
Screens

Project Name

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse molestie leo sem, id porttitor nunc ultricies dignissim. Nullam consectetur dictum nibh, sed volutpat nunc aliquet id. Curabitur libero diam, egestas in volutpat id, scelerisque et ipsum.

Before After

Screens



Project Name

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse molestie leo sem, id porttitor nunc ultricies dignissim. Nullam consectetur dictum nibh, sed volutpat nunc aliquet id. Curabitur libero diam, egestas in volutpat id, scelerisque et ipsum.

Before After

Screens