**TaskMaster - Todo Web Application - Project Documentation**

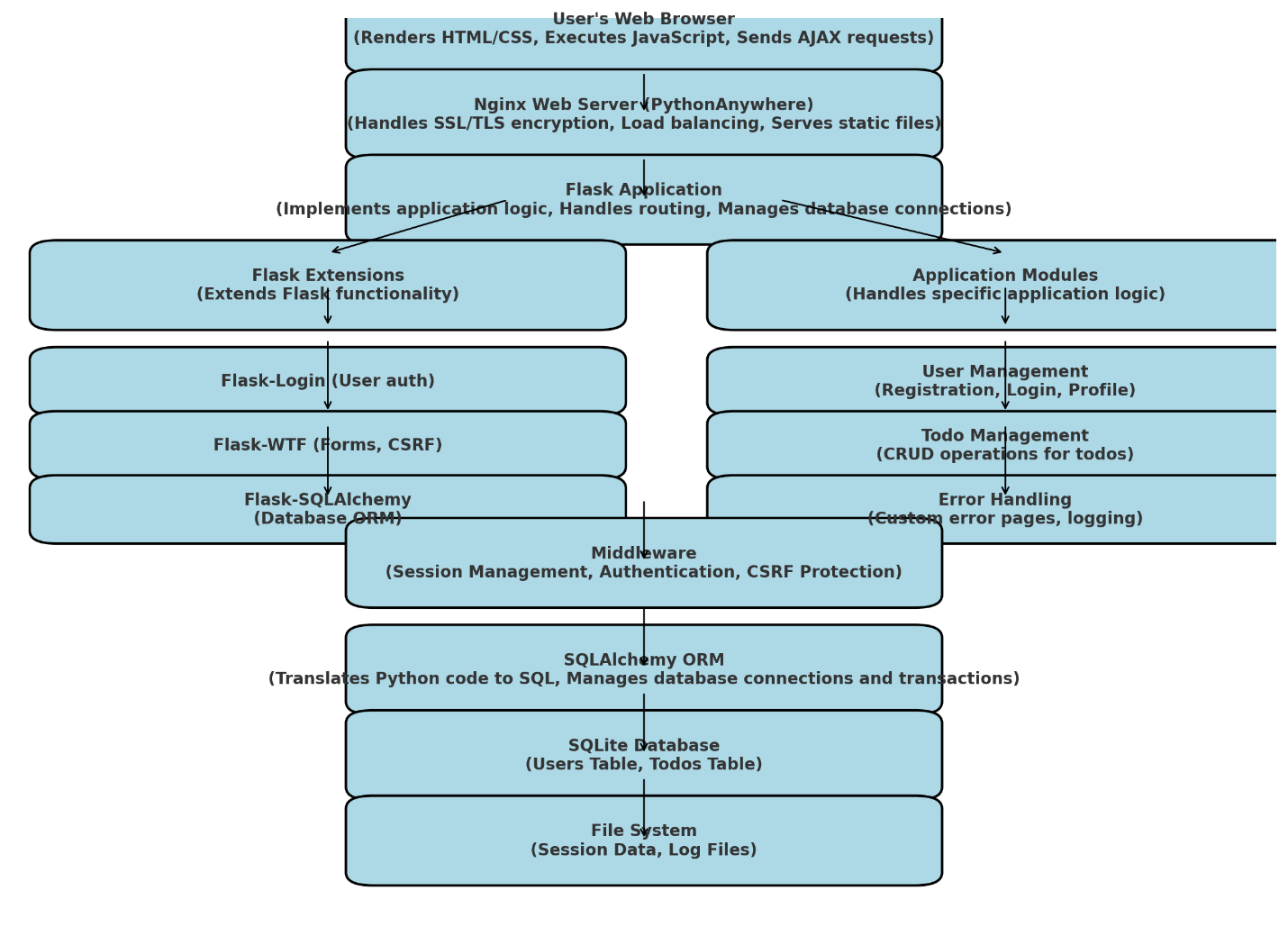
## **1. High-Level Description**

The Todo Web Application is a task management system designed to help users organize their daily activities efficiently. Built using Flask, a Python web framework, this application allows users to create, read, update, and delete tasks in a user-friendly interface. The app features user authentication, ensuring that each user's tasks remain private and secure.

Key features include:

* User registration and login
* Task creation with title and description
* Task completion tracking
* Task editing and deletion
* Persistent sessions for improved user experience

**2. Whiteboard Architecture Diagram**

****

### **Description of the Diagram**

The application follows a typical three-tier architecture:

1. Presentation Layer:
   * HTML/CSS/JavaScript for the frontend
   * Flask templates for dynamic content rendering
2. Application Layer:
   * Flask web server
   * Python business logic
   * Flask-Login for session management
3. Data Layer:
   * SQLite database
   * SQLAlchemy ORM for database interactions

### **Processes and Services**

1. Web Server (Flask):
   * Handles HTTP requests and responses
   * Routes requests to appropriate functions
2. Authentication Service:
   * Manages user registration, login, and logout
   * Utilizes Flask-Login for session handling
3. Task Management Service:
   * Handles CRUD operations for tasks
   * Interacts with the database through SQLAlchemy
4. Database Service (SQLite):
   * Stores user information and tasks
   * Provides data persistence

**3. Design Decisions and Justifications**

1. Choice of Flask Framework:
   * Lightweight and easy to set up
   * Provides necessary features without unnecessary complexity
   * Large community and extensive documentation
2. SQLite Database:
   * Serverless database, ideal for small to medium-scale applications
   * Easy to deploy and requires no additional setup
   * Suitable for applications with concurrent users but not extremely high traffic
3. SQLAlchemy ORM:
   * Abstracts database operations, making it easier to work with different databases if needed in the future
   * Provides a Pythonic way to interact with the database
4. Flask-Login for Session Management:
   * Simplifies user session handling
   * Integrates well with Flask and provides necessary security features
5. Filesystem Session Storage:
   * Allows for persistent sessions across app restarts
   * Simple to implement and suitable for the application's scale

**4. System Requirements and Testability**

**Functional Requirements:**

1. User Registration
   * Testable by creating new accounts and verifying database entries
2. User Authentication
   * Testable by logging in with correct and incorrect credentials
3. Task Creation
   * Testable by adding new tasks and verifying their appearance in the user's task list
4. Task Updating
   * Testable by modifying existing tasks and verifying changes
5. Task Deletion
   * Testable by removing tasks and confirming their absence from the task list
6. Task Completion Toggling
   * Testable by marking tasks as complete/incomplete and verifying status changes

**Non-Functional Requirements:**

* Performance
  + Response time should be under 2 seconds for all operations
  + Testable using performance testing tools like Apache JMeter
* Security
  + Passwords should be hashed before storage
  + Testable by inspecting database contents and attempting unauthorized access
* Usability
  + Interface should be intuitive and responsive
  + Testable through user testing and feedback
* Reliability
  + Application should handle errors gracefully
  + Testable by simulating various error conditions and observing system response
* Scalability
  + Should support up to 1000 concurrent users
  + Testable through load testing with tools like Locust

**5. Deployment and Access**

The application is deployed and accessible online at <https://aqsaakhan.pythonanywhere.com/>.

For demonstration purposes, you can use the following credentials:

* Username: aqsaanwar
* Password: 12345

The source code is available in the Git repository: https://github.com/aqsaakhan/ToDo-Webapp

**6. Future Improvements**

1. Implement task categories or tags for better organization
2. Add due dates and reminders for tasks
3. Develop a mobile app version for increased accessibility
4. Implement data backup and export features
5. Add user profile customization options

Aqsa Anwar