Smart To-Do List MVP Specification

Team:

1. Sheelah Mogaka

Rules and Justifications:

I will be designing and implementing the user interface, ensuring it is intuitive and user-friendly. I have knowledge and experience using Javascript, HTML, and CSS and I will be using them to make an engaging and accessible user experience.

I will also be handling the server-side development, ensuring the backend system is robust and scalable. I will be using Flask and SQLAlchemy, to ensure that the system can handle large volumes of data and user requests efficiently and thet the backend communicates well with the frontend.

Technologies:

- **Languages:** JavaScript, Python
- **Libraries/Frameworks:** Flask, SQLAlchemy
- **Platforms:** GitHub

Challenges:

Risks:

- **Technical Risks:**
    1. **Data Security:** Ensuring user data is secure is crucial. I will implement encryption and follow best practices for data protection.
    2. **Scalability:** The system must handle a large number of users without performance degradation. I plan to use scalable cloud infrastructure and optimize our code for performance.

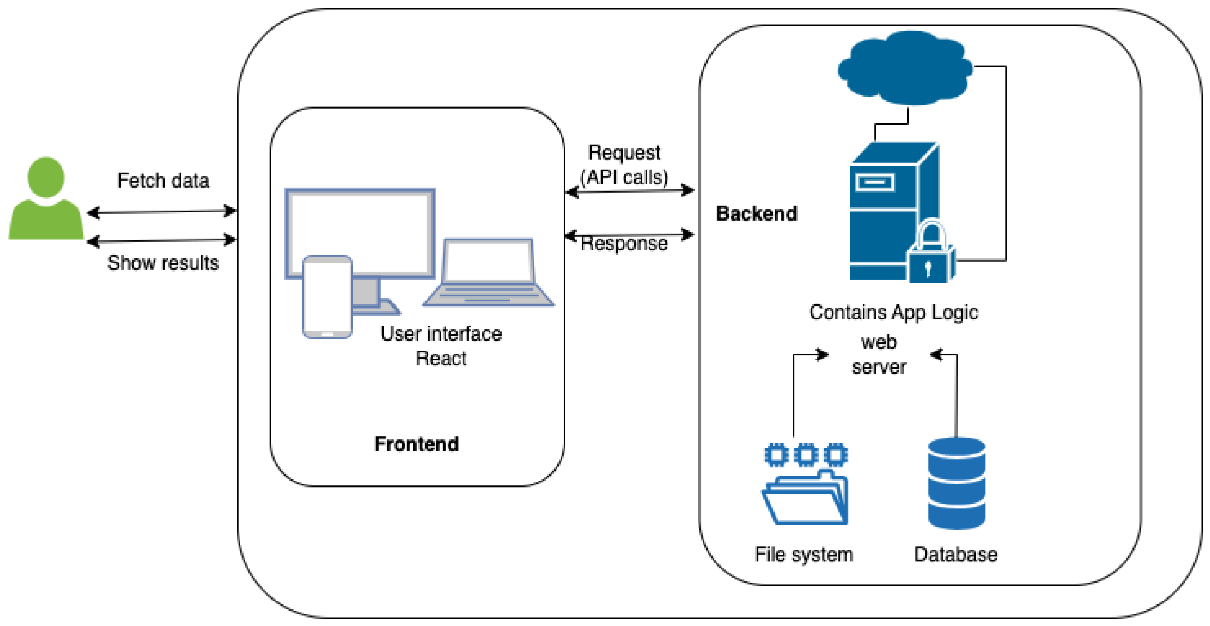
- **Non-Technical Risks:**
    1. **User Adoption:** Attracting users and ensuring they adopt the application can be challenging. I will focus on a user-friendly design and effective marketing strategies.
    2. **User Engagement:** Keeping users engaged over time is essential. I will include features like reminders and notifications to keep users actively using the app.

Infrastructure:

Existing Solutions:

- **Microsoft To Do:** Integrates with other Microsoft products and offers a simple interface for task management. I aim to provide a more intuitive and customizable user experience.
- **Trello:** Uses boards and cards for task management, suitable for project management. My solution will be simpler, focusing on personal to-do lists with smart features.

Architecture



APIs and Methods

API Routes

/api/tasks

- GET: Returns a list of all tasks for the authenticated user.
- POST: Creates a new task for the authenticated user.

/api/tasks/{task_id}

- GET: Returns details of a specific task.
- PUT: Updates a specific task
- DELETE: Deletes a specific task.

/api/user

GET: Returns the authenticated user's information.

Example Endpoints

1. **GET /api/tasks**
    - **Description**: Retrieves all tasks for the authenticated user.

POST /api/tasks

- **Description**: Creates a new task for the authenticated user.

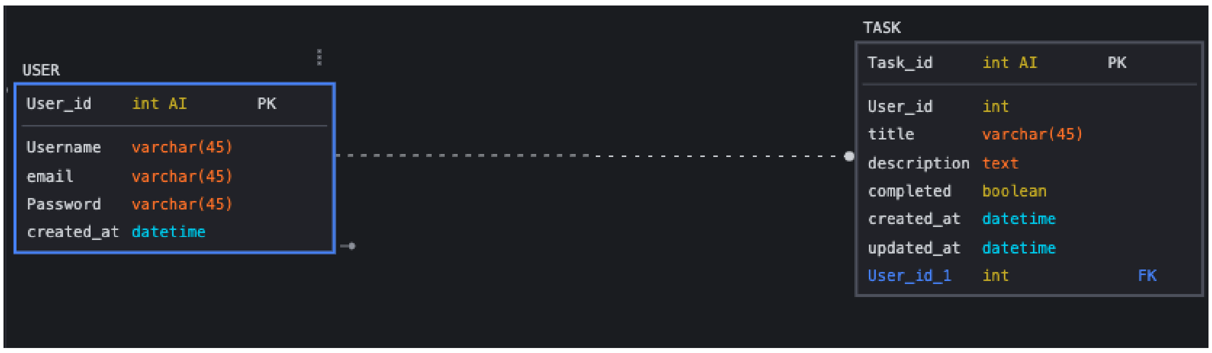
PUT /api/tasks/{task_id}

- **Description**: Updates an existing task.

DELETE /api/tasks/{task_id}

- **Description**: Deletes an existing task.

Data Modelling



User Stories

1. Task Creation

User Story:

As a busy professional, I want to create new tasks quickly and easily so that I can keep track of my to-do items without spending too much time on data entry.

Acceptance Criteria:

- The user can open a task creation form from the main interface.
- The user can enter a task title and description.
- The user can save the task, and it appears in the task list immediately.
- The form should validate that a title is provided.

2. Task Management

User Story:

As a task-oriented user, I want to view, edit, and delete tasks so that I can manage my tasks efficiently and keep my to-do list up to date.

Acceptance Criteria:

- The user can see a list of all their tasks.
- The user can click on a task to view its details.
- The user can edit the task's title and description.
- The user can delete a task, and it is removed from the list immediately.

3. Task Completion

User Story:

As a goal-driven individual, I want to mark tasks as completed so that I can track my progress and know which tasks are done.

Acceptance Criteria:

- The user can mark a task as completed with a single action.
- Completed tasks are visually distinct from incomplete tasks.
- The user can filter the list to show only completed, incomplete, or all tasks.

4. User Authentication

User Story:

As a security-conscious user, I want to sign up and login securely so that I can ensure that my to-do list is private and protected.

Acceptance Criteria:

- The user can create a new account with a username, email, and password.
- The user can log in with their email and password.
- Passwords are securely hashed and stored.
- The system provides feedback for incorrect login details.

5. Notifications

User Story:

As a forgetful user, I want to receive notifications for upcoming tasks so that I can be reminded to complete my tasks on time.

Acceptance Criteria:

- The user can set reminders for tasks.
- The system sends notifications at the specified reminder times.
- Notifications can be sent via email or push notifications, depending on user settings.