**Task Management System Project**

**The Idea:**

Our task management system is a collaborative web-based application designed to simplify task management for individuals. The system will allow users to create, assign, and track tasks easily and empower users to efficiently organize, and complete tasks, ensuring productivity. The system provides a user-friendly interface and a range of features like adding , deleting ,and updating tasks with due dates, and collaboration tools that will empower users to stay productive and organized.

**The problem Statement:**

Modern life is full of tasks and deadlines, which can be challenging to manage effectively. Many existing solutions have a lack of customization options, poor features, or intuitive design. This often leads to missed deadlines, unorganized processes, and decreased productivity**.** This project aims to solve these challenges by providing a user-friendly platform, and it will empower users to stay organized, meet deadlines, and enhance productivity through an efficient and user-friendly design.

**User Personas:**

**Sara**

* **Age:** 30
* **Occupation:** Software Engineer
* **Needs:** Manage work-related tasks and prioritize based on deadlines.
* **Pain Points:** Struggles with tracking project deadlines and assigning responsibilities in an unstructured task environment.

**Mohamed**

* **Age:** 22
* **Occupation:** University Student
* **Needs:** Organize study schedules, assignments, and part-time work tasks on a single platform.
* **Pain Points:** Finds it difficult to balance personal and academic commitments effectively.

**Ali**

* **Age:** 35
* **Occupation:** Entrepreneur
* **Needs:** Monitor team progress, categorize tasks by projects, and set reminders for important deadlines.
* **Pain Points:** Lacks a system to delegate and track tasks across multiple categories.

**Requirements:**

1. **Functional Requirements**

* **User Authentication:**
  + Register (Sign Up)
  + Sign in
  + Log out
* **Tasks Management** 
  + Create a Task (Title, description, due date)
  + Delete Task
  + Update Task
  + Mark the Task as Completed
  + Mark the Task as Incomplete
  + Categorize Tasks

1. **Non- Functional Requirements**
2. **Usability**

• The system should have an intuitive and user-friendly interface.

• The system should provide clear feedback to users for their actions.

1. **Performance**:

• The system should handle multiple users and tasks efficiently.

• The system should respond to user actions fast.

1. **Security**:

• User data should be securely stored and transmitted.

• The system should implement proper authentication and authorization

• mechanisms.

1. **Scalability:**

• The system should be able to accommodate a growing number of users and tasks.

1. **Reliability:**

• The system should be available 99.9% of the time.

• The system should handle errors and provide meaningful error messages

**Test Cases Outline**

|  |  |
| --- | --- |
| Test Case ID | Description |
| 1 | User Registration |
| 2 | User Registration with missing fields |
| 3 | User Login |
| 4 | User Login with invalid credentials |
| 5 | Add Task |
| 6 | Add Task with missing title |
| 7 | Update Task |
| 8 | Toggle Task completion |
| 9 | Delete Task |

**Test Case 1: User Registration**

Test Case ID:1

Description: Verify that a new user can register successfully.

Steps:

Navigate to the registration page (/register).

Enter a valid username and password.

Click the "Register" button.

Expected Result: The user should be redirected to the login page with a success message "Registration successful! Please log in."

**Test Case 2: User Registration with Missing Fields**

Test Case ID: 2

Description: Verify that the registration fails if the username or password is missing.

Steps:

Navigate to the registration page (/register).

Leave the username or password field empty.

Click the "Register" button.

Expected Result: The user should see an error message "Username and Password are required."

**Test Case 3: User Login**

Test Case ID: 3

Description: Verify that a registered user can log in successfully.

Steps:

Navigate to the login page (/login).

Enter a valid username and password.

Click the "Login" button.

Expected Result: The user should be redirected to the home page with a success message "Logout" visible.

**Test Case 4: User Login with Invalid Credentials**

Test Case ID: 4

Description: Verify that the login fails with invalid credentials.

Steps:

Navigate to the login page (/login).

Enter an invalid username or password.

Click the "Login" button.

Expected Result: The user should see an error message "Login failed. Check your username or password."

**Test Case 5: Add Task**

Test Case ID: 5

Description: Verify that a logged-in user can add a new task.

Steps:

Log in to the application.

Navigate to the home page.

Enter task details (title, description, due date, category).

Click the "Add Task" button.

Expected Result: The task should be added to the task list with a success message "Task added successfully!"

**Test Case 6: Add Task with Missing Title**

Test Case ID: 6

Description: Verify that adding a task fails if the title is missing.

Steps:

Log in to the application.

Navigate to the home page.

Leave the title field empty.

Click the "Add Task" button.

Expected Result: The user should see an error message "Title is required."

**Test Case 7: Update Task**

Test Case ID: 7

Description: Verify that a logged-in user can update an existing task.

Steps:

Log in to the application.

Navigate to the home page.

Click the "Edit" button for a task.

Update the task details.

Click the "Update Task" button.

Expected Result: The task should be updated with a success message "Task updated successfully!"

**Test Case 8: Toggle Task Completion**

Test Case ID: 8

Description: Verify that a logged-in user can toggle the completion status of a task.

Steps:

Log in to the application.

Navigate to the home page.

Click the "Complete" button for a task.

Expected Result: The task's completion status should be toggled with a success message "Task updated successfully!"

**Diagrams**

1. **Use Case**

**A diagram of a task

Description automatically generated**

**Use Case Diagram Outline**

1. **Actors:**
   * **User**
2. **Use Cases:**
   * **Sign Up**
   * **Sign In**
   * **Logout**
   * **Create Task**
   * **Delete Task**
   * **Update Task**
   * **Mark as Completed**
3. **Relationships:**
   * **Sign In → Logout (User must be signed in)**
   * **Create Task → Delete Task (Requires at least one task to delete)**
   * **Create Task → Update Task (Requires at least one task to update)**
   * **Create Task → Mark as Completed (Requires at least one task to mark)**
4. **Class Diagram**

A computer screen shot of a computer

Description automatically generated

**Class Diagram Outline**

1. **Classes:**
   * **User** 
     + **Attributes: username, password**
     + **Methods: signUp(), signIn(), logout()**
   * **Task** 
     + **Attributes: taskId, title, description, status**
     + **Methods: createTask(), deleteTask(), updateTask(), markAsCompleted()**
2. **Relationships:**
   * **User "1" --- "0..\*" Task (A user can have multiple tasks)**
   * **User "1" --- "1" Session (A user has one active session)**
3. **Sequence Diagram**

A diagram of a project

Description automatically generated with medium confidence

**Sequence Diagram Outline**

**Participants**

* **User**
* **TaskManagementSystem**
* **Task (for task-related operations)**

**Use Cases**

1. **User Sign Up**
2. **User Sign In**
3. **User Logout**
4. **Create Task**
5. **Delete Task**
6. **Update Task**
7. **Mark as Completed**

**Sequence Flow**

1. **User Sign Up**
   * **User → TaskManagementSystem: signUp(username, password)**
   * **TaskManagementSystem → User: confirmation**
2. **User Sign In**
   * **User → TaskManagementSystem: signIn(username, password)**
   * **TaskManagementSystem → User: sessionToken**
3. **User Logout**
   * **User → TaskManagementSystem: logout(sessionToken)**
   * **TaskManagementSystem → User: confirmation**
4. **Create Task**
   * **User → TaskManagementSystem: createTask(title, description)**
   * **TaskManagementSystem → Task: create(title, description)**
   * **Task → TaskManagementSystem: taskId**
   * **TaskManagementSystem → User: taskCreated**

**Task Management System**

**Frontend**

**• HTML: Used to structure the content of the web pages.**

**• CSS: Used for styling the web pages, including layout, colors, and fonts.**

**JavaScript: Used to add interactivity to the web pages, such as handling form submissions and toggling**

**task completion.**

**Backend**

**Python: The primary language used for the backend, specifically with the Flask framework to handle web requests, user authentication, and interaction with the database.**

**Database**

**• SQL: Used for defining and interacting with the database schema. In this project, SQLite is used as the database management system, which supports SQL for querying and managing the data.**

**These languages work together to create a full-stack web application, with the fronted handling the user interface and interactivity, the backend managing the application logic and server-side processing, and the database storing and retrieving data.**

Register (Sign Up)

A screenshot of a computer

Description automatically generated

Login

A screenshot of a computer

Description automatically generated

Home Page

A screenshot of a computer

Description automatically generated

“Dashboard”

Create Tasks (Title, description, due date)

A screenshot of a computer

Description automatically generated