**Yuossef hassan 221000750**

**Mohamed Osman 211001922**

**Mohamed tarek 221001435**

**Task Management Tool**

1. Introduction

This documentation covers a web-based Task Management Tool built with Python Flask backend and HTML/CSS/JavaScript frontend. The system allows users to register, log in, and manage their tasks through a dashboard interface.

2. System Overview

2.1 Technology Stack

- Frontend\*\*: HTML5, Tailwind CSS, JavaScript

- Backend\*\*: Python Flask

- Database\*\*: SQLite

- Authentication\*\*: JWT (JSON Web Tokens)

2.2 Key Features

- User registration and authentication

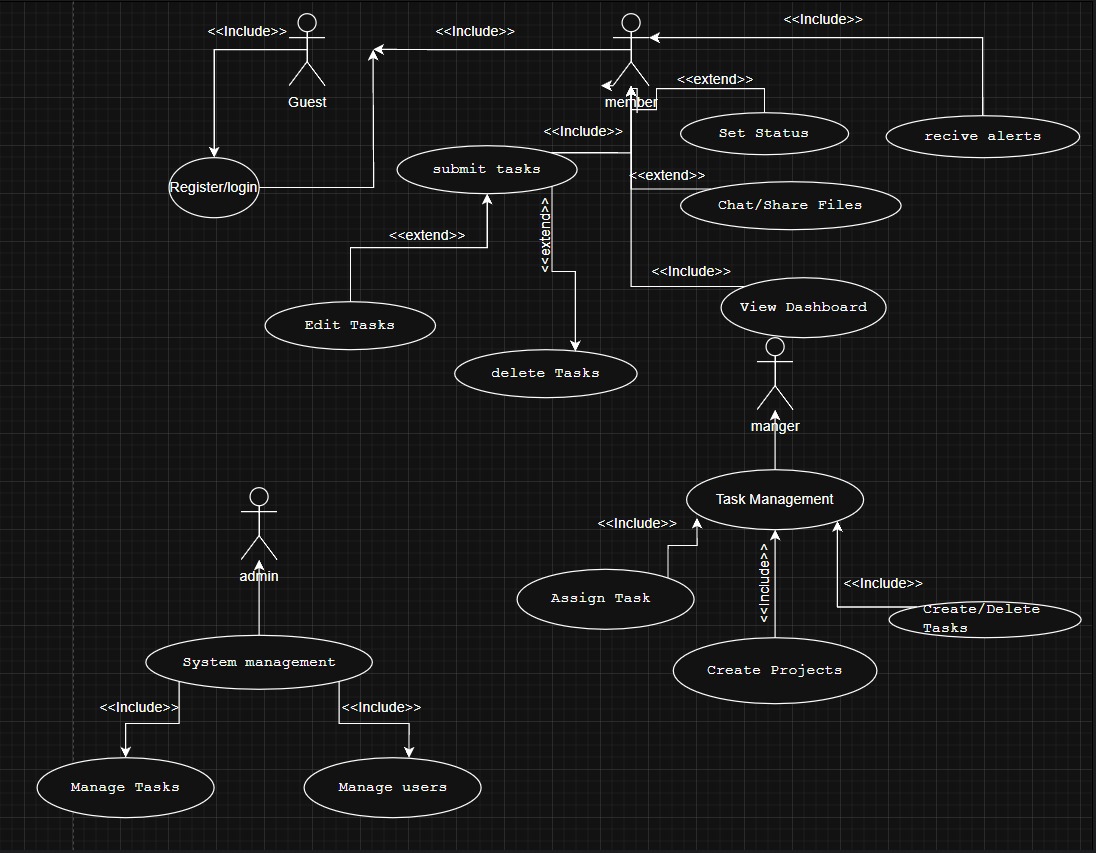
- Task creation and management

- Responsive design

- Secure session management

- RESTful API endpoints

3. Use Case Diagram



Use Case Descriptions

Actor Definitions

1. **Guest**
   * Unauthenticated user
   * Can register or log in to the system
2. **Member**
   * Regular authenticated user
   * Can perform basic task operations
   * Can collaborate with team members
3. **Manager**
   * Privileged user with additional permissions
   * Can manage team assignments and projects
   * Has administrative capabilities
4. **Admin** (implied from class diagram)
   * System administrator
   * Manages user accounts and system settings
   * Has full access to all features

Use Case Descriptions

Authentication Use Cases

1. **Register/Login** (Guest)
   * **Description**: Allows users to create accounts or authenticate
   * **Related Class**: User
   * **Preconditions**: None
   * **Postconditions**: User gains appropriate access level
2. **View Dashboard** (Member/Manager)
   * **Description**: Displays personalized task overview
   * **Related Classes**: User, Task, Project
   * **Preconditions**: Successful authentication
   * **Postconditions**: Dashboard rendered with user-specific data

Task Management Use Cases

1. **Submit Tasks** (Member)
   * **Description**: Create new task items
   * **Related Classes**: Task, User
   * **Preconditions**: Member logged in
   * **Postconditions**: New task appears in system
2. **Delete Tasks** (Member/Manager)
   * **Description**: Remove tasks from system
   * **Related Classes**: Task, Notification
   * **Preconditions**: User has appropriate permissions
   * **Postconditions**: Task removed and notifications sent if assigned
3. **Set Task Status** (Member/Manager)
   * **Description**: Update task progress state
   * **Related Classes**: Task, Notification
   * **Preconditions**: Task exists and is assigned to user
   * **Postconditions**: Task status updated, notifications triggered

Manager-Specific Use Cases

1. **Assign Tasks** (Manager)
   * **Description**: Delegate tasks to team members
   * **Related Classes**: Task, User, Notification
   * **Preconditions**: Manager logged in, task exists
   * **Postconditions**: Task assigned and notification sent
2. **Create Projects** (Manager)
   * **Description**: Establish new project containers
   * **Related Classes**: Project, User
   * **Preconditions**: Manager logged in
   * **Postconditions**: New project available for task organization

Collaboration Use Cases

1. **Chat/Share Files** (Member/Manager)
   * **Description**: Team communication and file exchange
   * **Related Classes**: Message, File, User
   * **Preconditions**: User authenticated
   * **Postconditions**: Messages/files stored in system

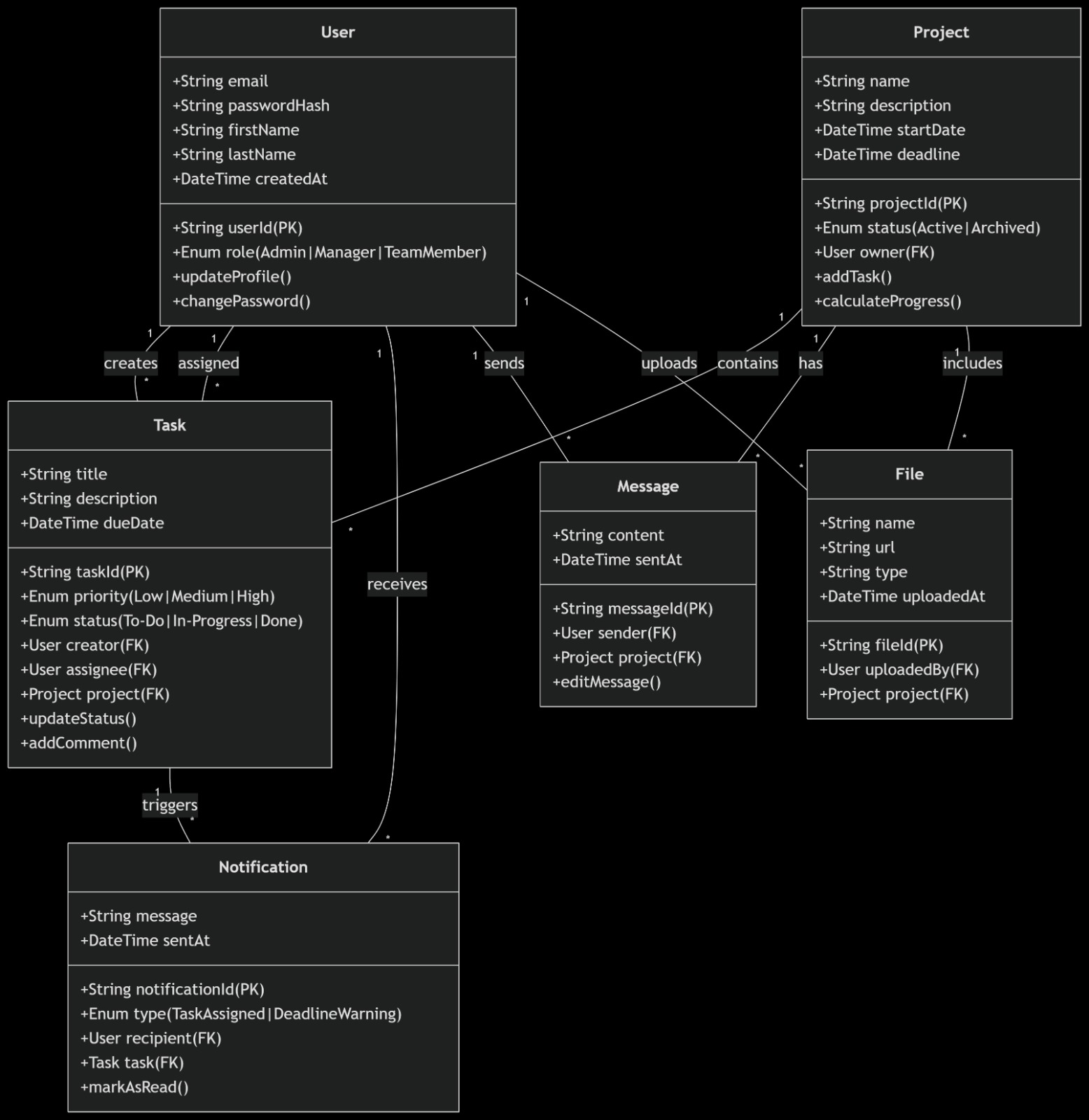
Admin Use Cases (implied)

1. **System Management** (Admin)
   * **Description**: Configure system settings and users
   * **Related Classes**: User
   * **Preconditions**: Admin privileges
   * **Postconditions**: System configuration updated

Relationships

1. **Inclusion Relationships** (<<include>>):
   * "Register/login" includes authentication verification
   * "Manage Tasks" includes status updates and assignments
   * "Task Management" includes all task-related operations
2. **Extension Relationships** (<<extend>>):
   * "Delete Tasks" extends from basic task management
   * "Set Status" extends from task operations
   * "System management" extends from user management

4. Class Diagram



1. User Class

**Attributes:**

* userId: Primary key (String)
* email: Unique user identifier (String)
* passwordHash: Securely stored password (String)
* firstName: User's first name (String)
* lastName: User's last name (String)
* createdAt: Account creation timestamp (DateTime)
* role: User role (Enum: Admin|Manager|TeamMember)

**Methods:**

* updateProfile(): Updates user information
* changePassword(): Changes user password

**Relationships:**

* Creates Tasks (1-to-many)
* Receives Notifications (1-to-many)
* Owns Projects (1-to-many)
* Uploads Files (1-to-many)
* Sends Messages (1-to-many)

2. Task Class

**Attributes:**

* taskId: Primary key (String)
* title: Task name (String)
* description: Task details (String)
* dueDate: Completion deadline (DateTime)
* priority: Importance level (Enum: Low|Medium|High)
* status: Completion state (Enum: To-Do|In-Progress|Done)

**Methods:**

* updateStatus(): Changes task status
* addComment(): Adds comments to task

**Relationships:**

* Belongs to Project (many-to-1)
* Created by User (many-to-1)
* Assigned to User (many-to-1)
* Triggers Notifications (1-to-many)

3. Notification Class

**Attributes:**

* notificationId: Primary key (String)
* message: Notification content (String)
* sentAt: Creation timestamp (DateTime)
* type: Notification category (Enum: TaskAssigned|DeadlineWarning)

**Methods:**

* markAsRead(): Updates read status

**Relationships:**

* Sent to User (many-to-1)
* Related to Task (many-to-1)

4. Project Class

**Attributes:**

* projectId: Primary key (String)
* name: Project title (String)
* description: Project details (String)
* startDate: Initiation date (DateTime)
* deadline: Completion date (DateTime)
* status: Current state (Enum: Active|Archived)

**Methods:**

* addTask(): Creates new tasks
* calculateProgress(): Computes completion percentage

**Relationships:**

* Owned by User (many-to-1)
* Contains Tasks (1-to-many)
* Contains Files (1-to-many)
* Contains Messages (1-to-many)

5. File Class

**Attributes:**

* fileId: Primary key (String)
* name: File name (String)
* url: Storage location (String)
* type: File format (String)
* uploadedAt: Creation timestamp (DateTime)

**Relationships:**

* Uploaded by User (many-to-1)
* Belongs to Project (many-to-1)

6. Message Class

**Attributes:**

* messageId: Primary key (String)
* content: Message text (String)
* sentAt: Creation timestamp (DateTime)

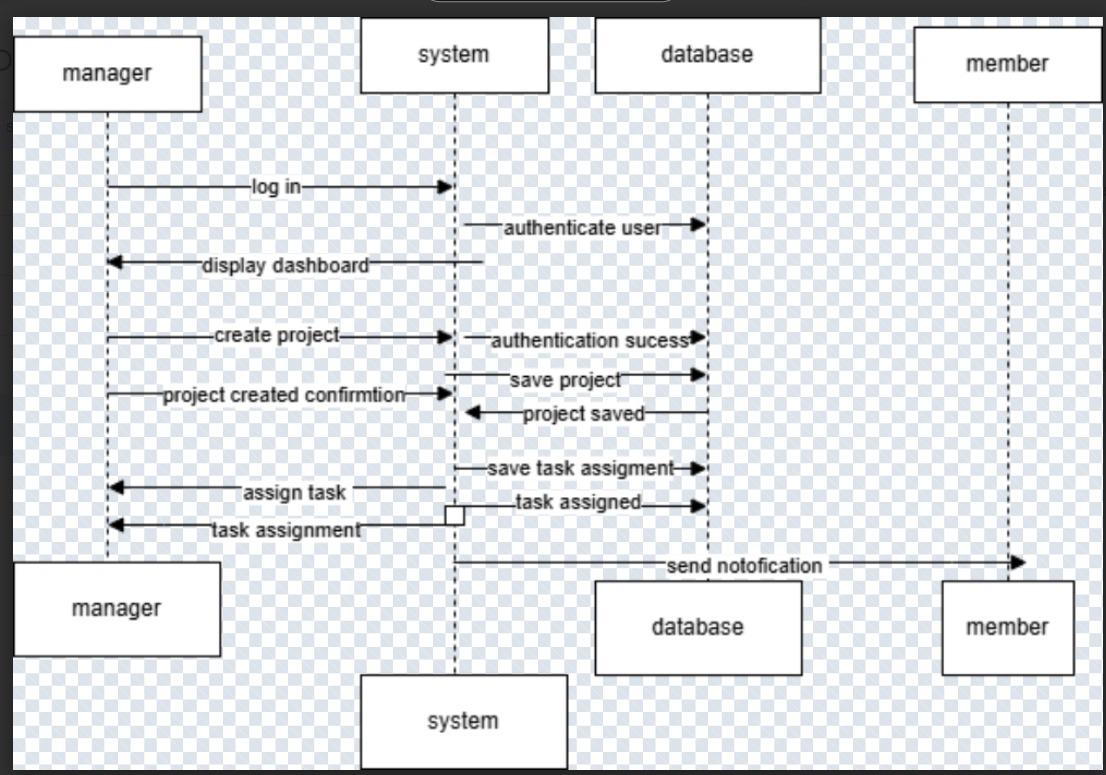
**Methods:**

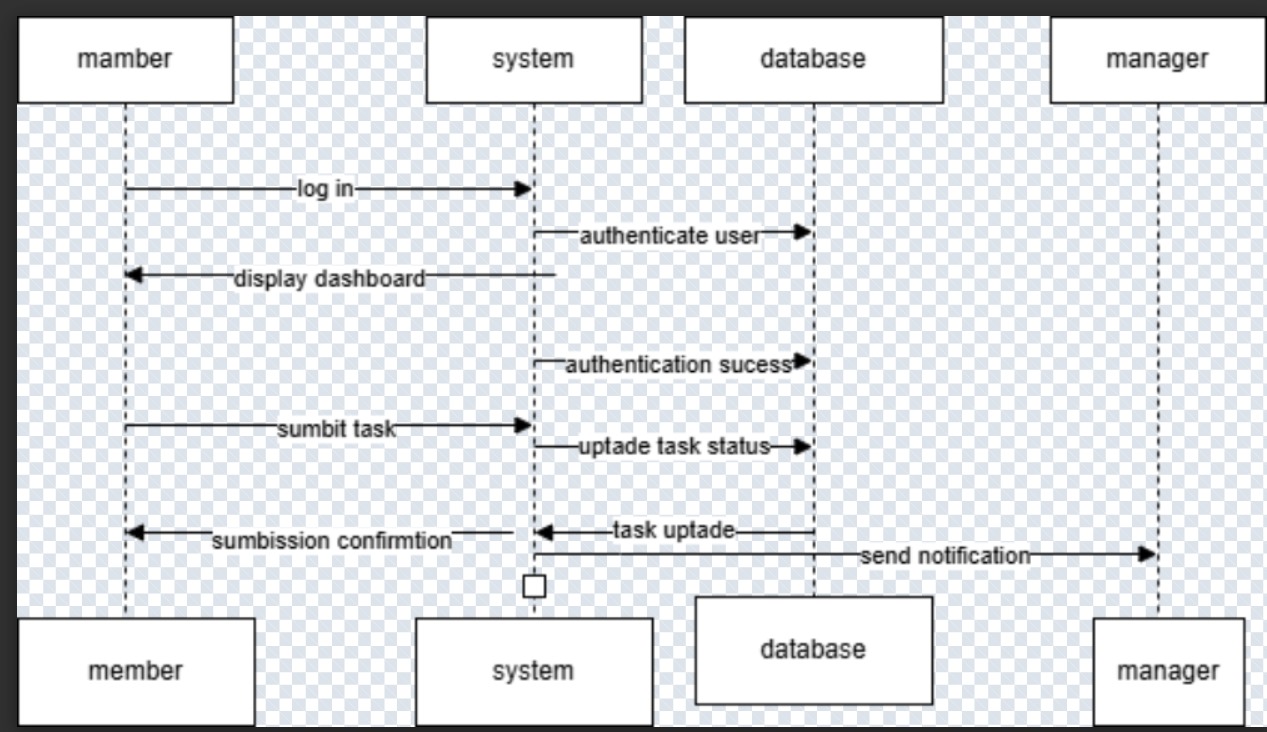
* editMessage(): Modifies message content

**Relationships:**

* Sent by User (many-to-1)
* Belongs to Project (many-to-1)

5. Sequence Diagrams





5.1 User Registration

```plaintext

User -> Browser: Fills registration form

Browser -> Flask: POST /api/register (form data)

Flask -> Database: Create new user

Database -> Flask: User record

Flask -> Browser: 201 Created + JWT

Browser -> LocalStorage: Store token

Browser -> User: Redirect to dashboard

```

5.2 User Login

```plaintext

User -> Browser: Fills login form

Browser -> Flask: POST /api/login (credentials)

Flask -> Database: Validate credentials

Database -> Flask: User record

Flask -> Browser: 200 OK + JWT

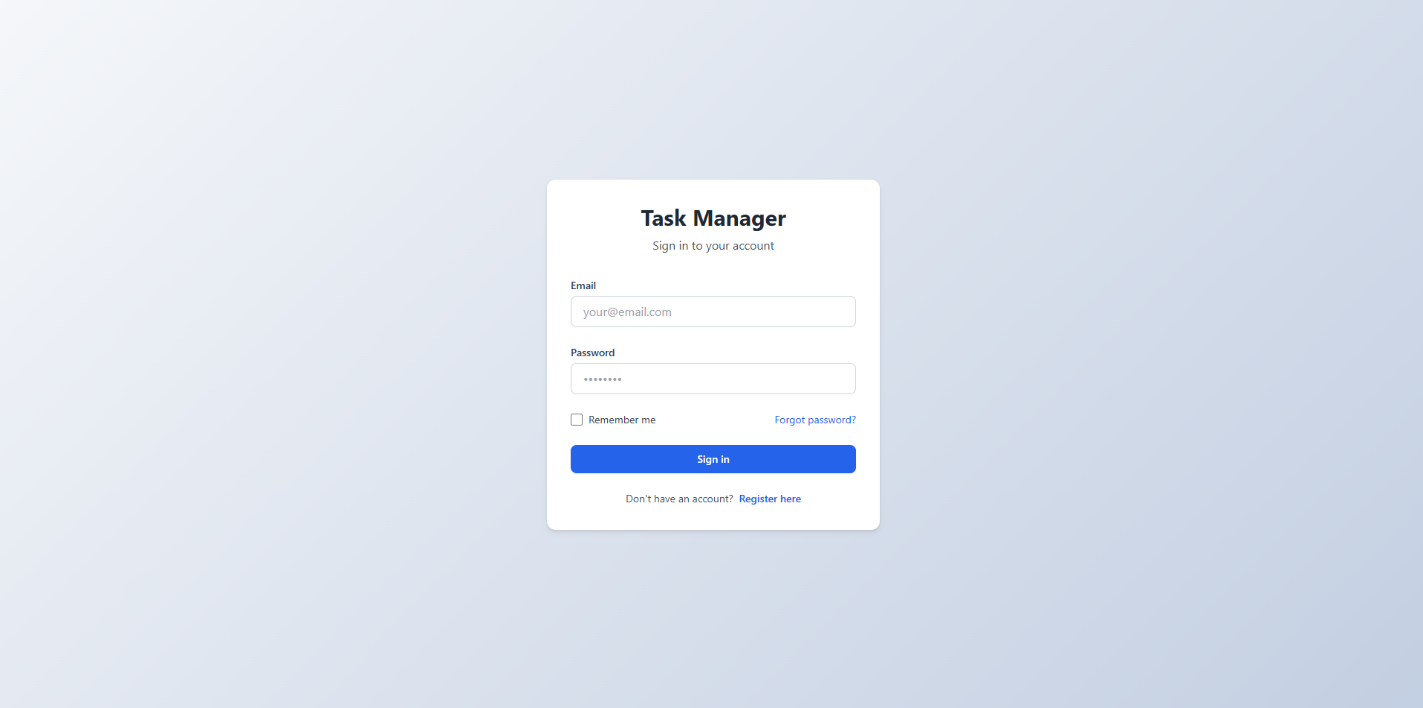
Browser -> Storage: Store token (local/session)

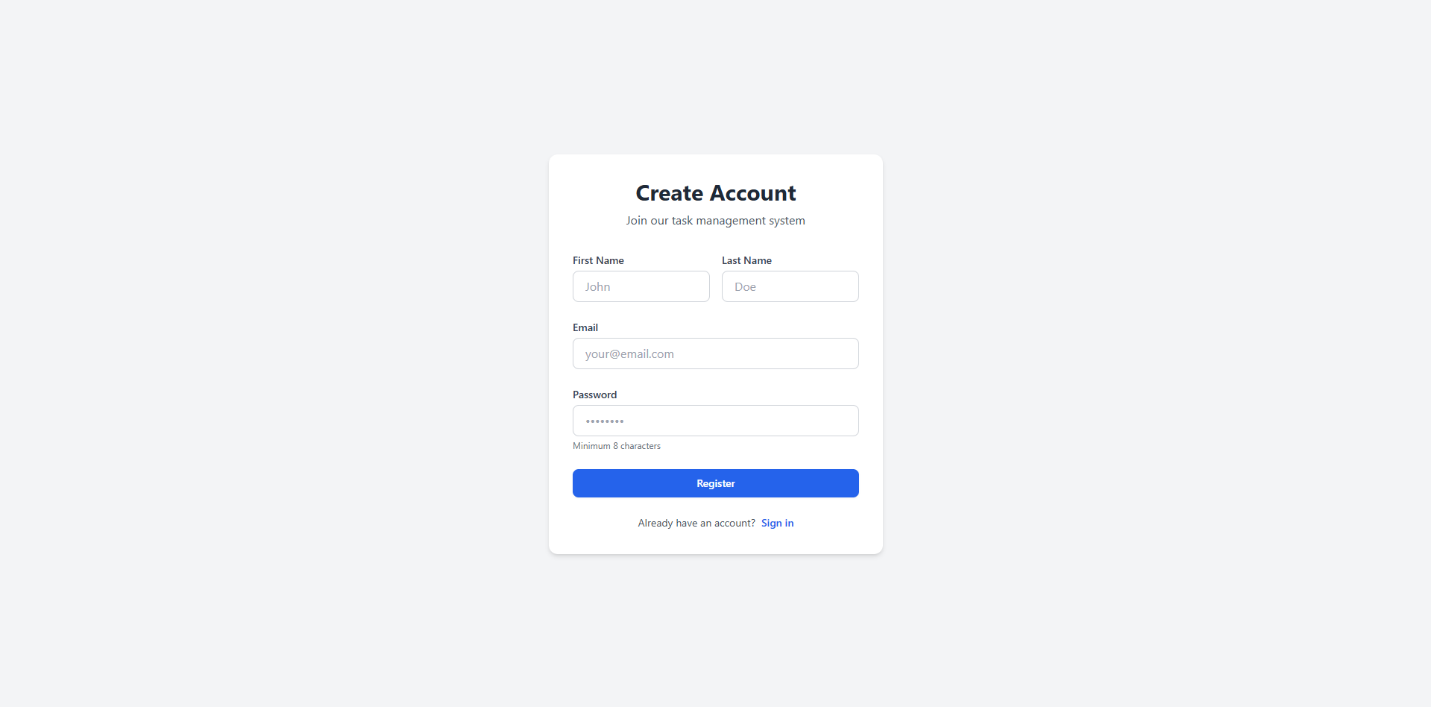
Browser -> User: Redirect to dashboard

```

5.3 Task Management

6. Frontend Components





1. Login Page

- Email/password input

- Remember me option

- Registration link

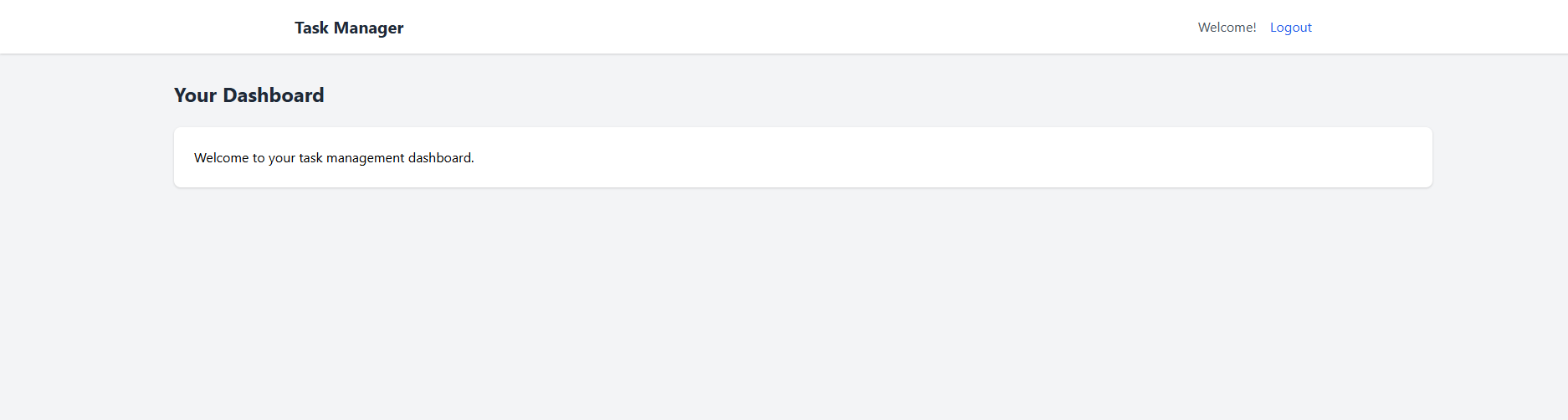
2. Registration Page

- Name, email, password fields

- Form validation

- Password requirements

3. Dashboard



- Task overview statistics

- Task list with completion toggle

- Logout functionality

7. Conclusion

This Task Management Tool provides a secure, responsive web interface for personal task organization. The system follows modern web development practices with a clear separation between frontend and backend components, implemented with Flask and Tailwind CSS. The documentation covers all aspects of the system from use cases to technical implementation details.

**Chapter 6: Testing**

**6.1 Test Cases**

| **Test Case ID** | **Description** | **Expected Result** | **Actual Result** | **Status** |
| --- | --- | --- | --- | --- |
| TC01 | User registers with valid credentials | Redirect to dashboard with success message | As expected | ✅ Pass |
| TC02 | Login with incorrect password | Show error message, no login | As expected | ✅ Pass |
| TC03 | Create a new task with valid data | Task appears in dashboard | As expected | ✅ Pass |
| TC04 | Delete a task as a member | Task is removed from the system | As expected | ✅ Pass |
| TC05 (Negative) | Try submitting a task with blank title | Show form validation error | As expected | ✅ Pass |

**Chapter 7: Risk and Security Requirements**

**7.1 Risk Identification**

| **Risk ID** | **Category** | **Description** |
| --- | --- | --- |
| R1 | Security | Unauthorized access via token leakage |
| R2 | Performance | System slows down with 1000+ tasks |
| R3 | Usability | Users may forget to logout on shared devices |
| R4 | Data Integrity | Accidental deletion of tasks without undo |
| R5 | Hosting/Infra | Downtime due to server outage or database corruption |

**7.2 Risk Assessment**

| **Risk ID** | **Likelihood** | **Impact** | **Risk Level** |
| --- | --- | --- | --- |
| R1 | High | High | Critical |
| R2 | Medium | Medium | Moderate |
| R3 | Medium | Low | Low |
| R4 | Medium | High | High |
| R5 | Low | High | Moderate |

**7.3 Mitigation Techniques**

* **R1:** Implement token expiry and refresh mechanisms, secure HTTPS-only cookies.
* **R2:** Use task pagination or lazy loading.
* **R3:** Add auto-logout after inactivity.
* **R4:** Include a “Recycle Bin” or undo feature for deleted tasks.
* **R5:** Use cloud-based hosting with automated daily backups.

**7.4 Requirement Refinement (SRS Updates)**

**Functional Requirements**

* FR9: System shall implement a "soft delete" mechanism for tasks.
* FR10: System shall auto-logout users after 15 minutes of inactivity.

**Non-Functional Requirements**

* NFR5: System shall support up to 2000 concurrent tasks with <500ms response time.
* NFR6: All JWT tokens must be encrypted and stored securely in session cookies.