A white brain with arrows and dots

Description automatically generated

TRASKIO

CSCI313: Software Engineering

Under the supervision of Dr. Mourad Raafat

Nada Gamal Zaki – 221000829

Abdelrahman Alsammany – 221000915

Omar Hassan – 221000998

Yasser Mohamed – 221001983

Omar Ashraf – 221000793

Mohamed Eltwapty – 221001161

## **Introduction**

### **Overview**

The Task Tracking Tool is a productivity-focused application designed to simplify task organization and progress tracking. By offering features such as real-time updates, task categorization, and progress visualization, the tool provides users with an efficient platform for managing their tasks.

### **Purpose**

The primary purpose of the Task Tracking Tool is to enable individuals to organize, track, and manage their tasks effectively. With its intuitive design and practical features, the tool aims to enhance personal productivity and task prioritization.

### **Key Features**

* **Task Management:** Create, edit, and delete tasks with ease.
* **Progress Tracking:** Monitor task completion using dynamic progress bars.
* **Real-Time Updates:** Reflect changes instantly for an uninterrupted user experience.
* **Search and Filter:** Quickly locate tasks or mini tasks using titles, descriptions, or status.
* **Mini-Tasks:** Break down larger tasks into smaller, manageable sub-tasks with independent tracking.
* **Saved Data:** Ensure task information is persistently stored for continued access.
* **Task Categorization:** Filter tasks by status, such as "Completed," "In Progress," or "All Tasks."

### **Target Audience**

* **Students**: Ideal for students who need to organize and manage their academic assignments, projects, and study schedules. The tool helps them stay on top of deadlines, track progress, and break larger tasks into manageable mini tasks.
* **Professionals**: Suitable for professionals who need a simple yet powerful way to track their daily tasks, prioritize work, and maintain productivity.
* **Freelancers**: Freelancers can use the tool to manage their various client tasks and personal projects efficiently without the need for complex project management systems.
* **Home Managers**: Individuals responsible for managing household tasks, grocery shopping lists, or personal projects can also benefit from using the tool to stay organized and on track.
* **Hobbyists and Enthusiasts**: People who engage in personal hobbies or side projects can use the tool to break their activities into smaller tasks, set deadlines, and track progress.

The tool's simple interface and intuitive design make it accessible to users with varying technical skills, ensuring that anyone seeking to enhance their task management can use it effectively.

# **Functional Requirements:**

## **User Authentication & Account Management:**

* **Sign Up**: The user must be able to sign up using an **email address**, **password**, and **username**.
* **Login**: The user must be able to log in using either their **username** or **email** and p**assword**.

## **Homepage:**

* **Active Homepage**: The app should display a homepage that shows the **current time** and updates in real-time.
* The homepage should be the landing page after logging in.

## **Task Management:**

* **Add To-Do Card:** The user must be able to create a new To-Do card with a title.
* **Delete To-Do Card:** The user should be able to delete any created To-Do card.
* **Change To-Do Title:** The user should be able to edit the title of any existing To-Do card.
* **Add Mini Tasks:** The user should be able to add mini tasks within each To-Do card.
* **Check Mini Tasks:** The user should be able to mark mini tasks as completed.
* **Delete Mini Tasks:** The user should be able to remove any mini task.
* **Update Progress Bar:** A progress bar should dynamically update for each To-Do card based on the completion status of its mini tasks.

## **Search and Filter:**

* **Search Bar**: The user should be able to search for **To-Do cards** or **mini tasks** by title or description.
* **Filtering**: The user can filter tasks based on their status:
* **Done** tasks.
* **In Progress** tasks.
* **All** tasks (default view).

## **Non-Functional Requirements**

### **Performance**

* **Reasonable Loading Time:** The app should load within 3–5 seconds when navigating between pages, depending on network conditions.
* **Timely Updates:** Changes to tasks (e.g., checking mini tasks or deleting cards) should be reflected in under 1 second on most devices.

### **Scalability**

* **Moderate User Base:** The app should support up to 100 concurrent users with smooth performance.
* **Task Volume:** The app should handle up to 500 tasks per user without noticeable delays in loading or updates.

### **Usability**

* **Simple and Intuitive UI:** The interface should be straightforward to use, even for non-technical users, with clear navigation and labels.
* **Basic Accessibility:** The app should include basic accessibility features, such as keyboard navigation for core actions and proper screen reader compatibility for essential components.

### **Security**

* **Password Protection:** User passwords should be securely hashed using a reliable algorithm (e.g., bcrypt).
* **Session Management:** Maintain user state securely across pages during active sessions.
* **Data Transmission:** Use secure HTTPS protocols to protect user data during transmission.

### **Reliability**

* **High Availability:** The app should aim for at least 95% uptime, with planned downtime for updates and maintenance communicated in advance.
* **Data Recovery:** Task data should be saved locally in the browser or on the server to avoid accidental data loss during typical usage.

### **Consistency**

* **Visual Design:** Maintain a uniform layout across all pages, including headers, footers, and core components.
* **Color Scheme:** Use a cohesive color palette, ensuring good contrast for readability and avoiding colors like green and yellow except for specific elements.
* **Typography:** Ensure consistent font usage for headings, body text, and labels.
* **Button and Icon Design:** Keep buttons and icons consistent in style, color, and interaction feedback (e.g., hover effects).
* **Spacing and Alignment:** Ensure uniform spacing and alignment for a clean, organized look.
* **Functional Behavior:** Core features, such as task creation and progress tracking, should work consistently on supported devices and browsers.

### **Maintainability**

* **Modular Codebase:** Organize the codebase into clearly defined components (e.g., task cards, progress bar, header).
* **Basic Documentation:** Provide simple comments and documentation to help maintain and update the app efficiently.

1. **Tools and Technologies**
   1. **Frontend:**

* **Framework: React.js – React.js will be used to build the frontend, providing a responsive, interactive, and modern user interface.**
* **Language: JavaScript – JavaScript will be used for frontend logic and interaction with the backend APIs.**
  1. **Backend:**
* **Framework: Django** **– Django will power the backend, providing a robust framework for developing APIs and managing system logic.**
* **Language: Python** **will be used in conjunction with Django for server-side development.**
* **Database: PostgreSQL – PostgreSQL will be the relational database used for managing tasks, notes, and user information.**
* **Database Hosting: Neon Serverless PostgreSQL.**

## **Version Control and Collaboration:**

### **Tool: Git**

**Git will be used for version control, tracking changes to the codebase and enabling collaboration.**

* 1. **Development Tools:**
* **Tool: Visual Studio Code (VS Code)**

**Visual Studio Code will be used as the primary code editor, providing a development environment with extensions for React.js, Python, Django, and Git integration.**

# **Test Cases**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Description** | **Input Data** | **Expected Result** | **Result** | | **Status** |
| **T-01** | Login with valid credentials | Valid email and password | User successfully logs in | Successfully logged in | Pass | |
| **T-02** | Login with invalid credentials | Invalid email and password | Error message “Invalid credentials” | Displayed “Invalid Credentials” | Pass | |
| **T-03** | Create a new card | Card title: “Untitled Project” Progress bar: “0%” | New card is successfully created | Card created successfully | Pass | |
| **T-04** | Create a new task | Task title | New task is successfully created and added to a list | Task created successfully and available in the list | Pass | |
| **T-05** | Mark task as done | Completed task | Task state is updated to “Done” and progress bar updated | Task state and progress bar updated successfully | Pass | |
| **T-06** | Delete task | Deleted task | Task deleted form the list and progress bar updated | Task deleted and progress bar updated successfully | Pass | |
| **T-07** | Delete card | Deleted card | Card deleted | Card deleted successfully | Pass | |

# **Risk Analysis**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk** | **Description** | **Likelihood** | **Impact** | **Mitigation Strategy** |
| **Service Interruptions** | The application may experience downtime due to issues with the server or database infrastructure. | Medium | Critical | Utilize cloud-based solutions with automatic failover and redundancy to minimize downtime. |
| **Overloading System Resources** | System performance degradation when handling many concurrent users | Medium | High | Optimize codebase for better resource management, conduct load testing, and ensure scalability with cloud services. |
| **Integration Failures** | Issues with connecting services like AI features could cause disruptions. | Low | Medium | Conduct thorough testing of all integrations and ensure proper fallback mechanisms for third-party service failures. |
| **User Interface Issues** | Poor user interface design or unexpected user behaviors might reduce the app's usability. | High | Moderate | Conduct comprehensive user testing, gather feedback regularly, and refine the design to ensure it remains intuitive and user-friendly. |
| **User Experience Problems** | Complex or unintuitive UI reducing app usability. | High | Moderate | Gather user feedback actively, improve the design iteratively, and provide onboarding materials for new users. |
| **Version Control Conflicts** | Development team members might face conflicts when merging changes | Medium | Low | Establish clear Git workflow guidelines, encourage frequent pull requests, and conduct regular code reviews to resolve issues early. |

1. **User Interface**

The **Task Management Tool** offers a simple, intuitive, and user-friendly interface, designed to streamline task management while ensuring a smooth user experience.

* 1. **Homepage:**

The homepage displays the **current time** and presents the main features available for users. It also includes easy access to **Sign Up** or **Login** options for new or returning users.

A screenshot of a computer

Description automatically generated

* 1. **Sign Up Page:**

**Input Fields**: Users will be asked to provide: Username, Email, Password, Confirm Password.

**User Navigation:** If the user already has an account, they can navigate to the **Login** page via a link.

**Validation Alerts:**

* **Email Validation:** An alert will appear if the email provided does not include an "@" symbol:  
  alert('Enter a valid email address');
* **Password Matching:** An alert will appear if the password and confirm password fields do not match:  
  alert('Passwords do not match');

**A screen shot of a login form

Description automatically generated**

* 1. **Login Page**

**Input Fields:** Users can log in with either their username or email address associated with the account and password.

**User Navigation:** If the user doesn’t have an account, they can navigate to the Sign Up page via a link.

**Validation Alerts:**

* **Invalid Credentials:** If the login credentials are invalid, an alert will notify the user:  
  alert('Invalid credentials. Please try again');

**A screenshot of a login screen

Description automatically generated**

* 1. **ToDo Page (Main App):**

**A screenshot of a computer program

Description automatically generated**

* **Add Task:**  
  Users can click a "+" button to add a new To-Do card and provide a title for the task.
* **Add Mini Tasks:**  
  Users can add mini tasks inside each To-Do card, breaking down larger tasks into smaller, actionable steps.
* **Task Completion:**  
  Users can check off individual mini tasks as complete, which will automatically update the progress bar for that card.
* **Delete Task:**  
  Users can delete a mini task, and the progress bar will be updated accordingly.
* **Delete Card:**  
  Users can delete the entire To-Do card.

**A screenshot of a computer

Description automatically generated**

* 1. **Filter Options:**

**Filter by Task Status:** Users can filter tasks based on the following categories:

* **All Tasks (Default View):**  
  Displays all tasks, including both ongoing and completed tasks, as the default view.
* **In Progress:**  
  Displays tasks that are currently being worked on. Tasks that are not marked as completed will be listed under this category.
* **Done:**  
  Displays tasks that have been marked as completed. Users can track their progress by reviewing the tasks that are finished.

A screen shot of a computer

Description automatically generated

* 1. **Search Feature**

**Search by Card Title or Mini Task Title:** Users can search for tasks in the following ways:

* **Card Title:**  
  Users can enter keywords or specific phrases from the title of a To-Do card. This allows them to quickly locate a task card, especially when managing a large number of tasks.
* **Mini Task Title:**  
  Users can also search for specific mini tasks within a card by typing keywords or mini task titles. This feature makes it easier to focus on a specific subtask, especially when a card contains multiple mini tasks.

**Search with Filters:** The search functionality is enhanced by the filter options. Users can apply filters such as **All Tasks**, **In Progress**, or **Done** while searching, allowing them to narrow their search results to only the tasks that match both the selected filter and the search term. This integrated search and filter system improves task management, making it faster and more efficient for users to find and focus on specific tasks.

A screenshot of a computer program

Description automatically generated

* 1. **Profile Page**

The **Profile Page** allows users to view and update their personal information to keep their profile up to date while maintaining security for sensitive data like their email and password.

* **Username:**

Users can change their username at any time, allowing them to personalize their profile as needed.

* **Email:**

The email address is fixed and cannot be changed once registered to ensure account security.

* **Password:**

Users can update their password by entering their **old password**, followed by the **new password** and **confirmation of the new password**. This ensures that only the account owner can update the password.

* **Profile Picture:**

Users have the option to upload a custom profile picture. They can choose an image from their device and update their profile with a personal touch.

**A screenshot of a computer screen

Description automatically generated**

* 1. **Header**

The **Header** is fixed at the top of all pages and displays different options depending on the page you're on.

* **Homepage:** Displays the logo and options for **Sign Up** or **Login**.



* **Login/Sign Up Pages:** Includes a link to navigate back to the **Homepage**.



* **Todo Page (Main App):**
* **Search Option**: Allows users to search tasks by card title or mini task title.
* **Filter Option**: Users can filter tasks by **All**, **In Progress**, or **Completed**.
* **Account Options**: Includes links to the **Profile Page** and **Logout** options.



* **Profile Page**: Includes a link to the **Dashboard**.



# **Diagrams**

## **Class Diagram:**

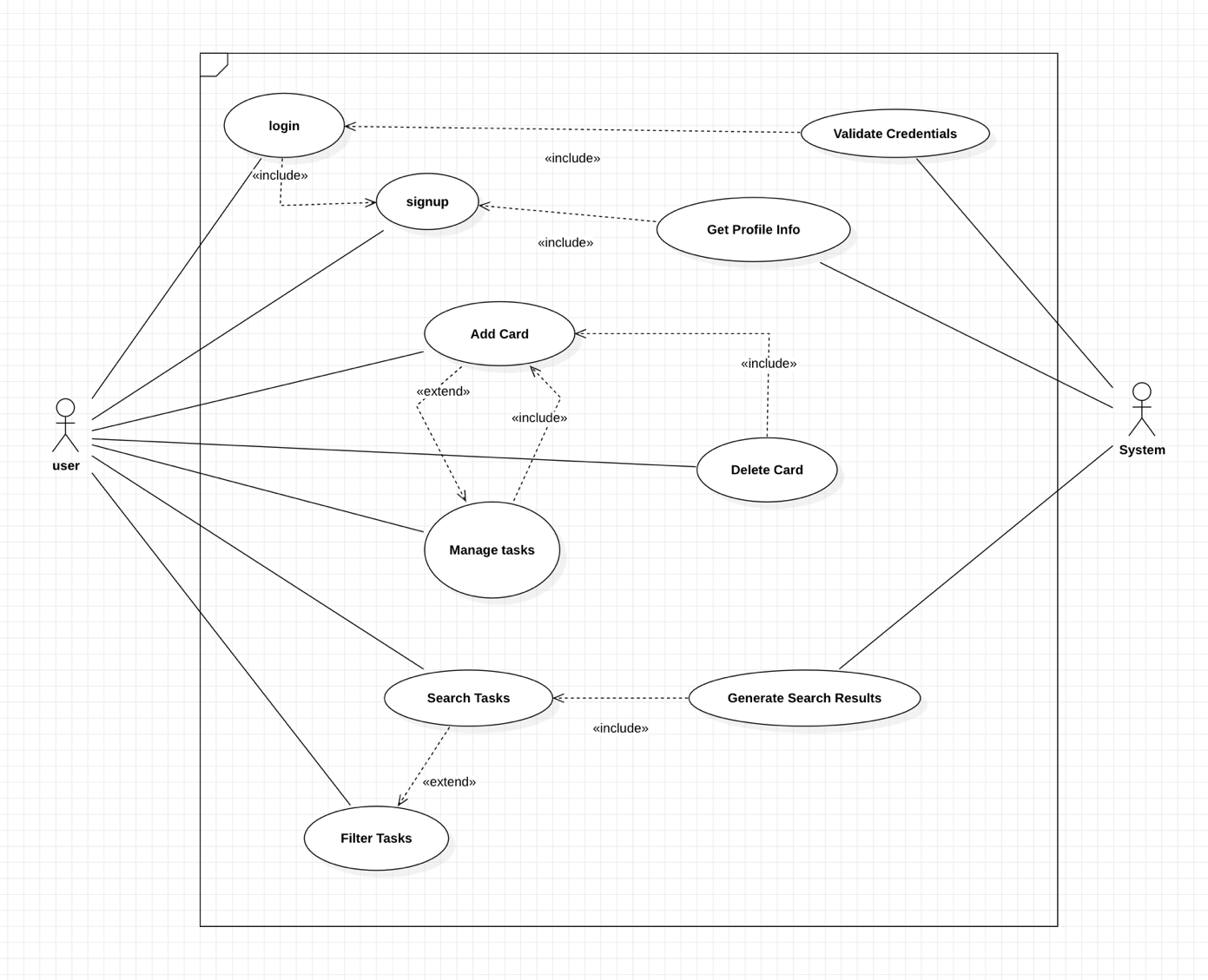
## **A diagram of a computer Description automatically generated**

## **Sequence Diagram:**

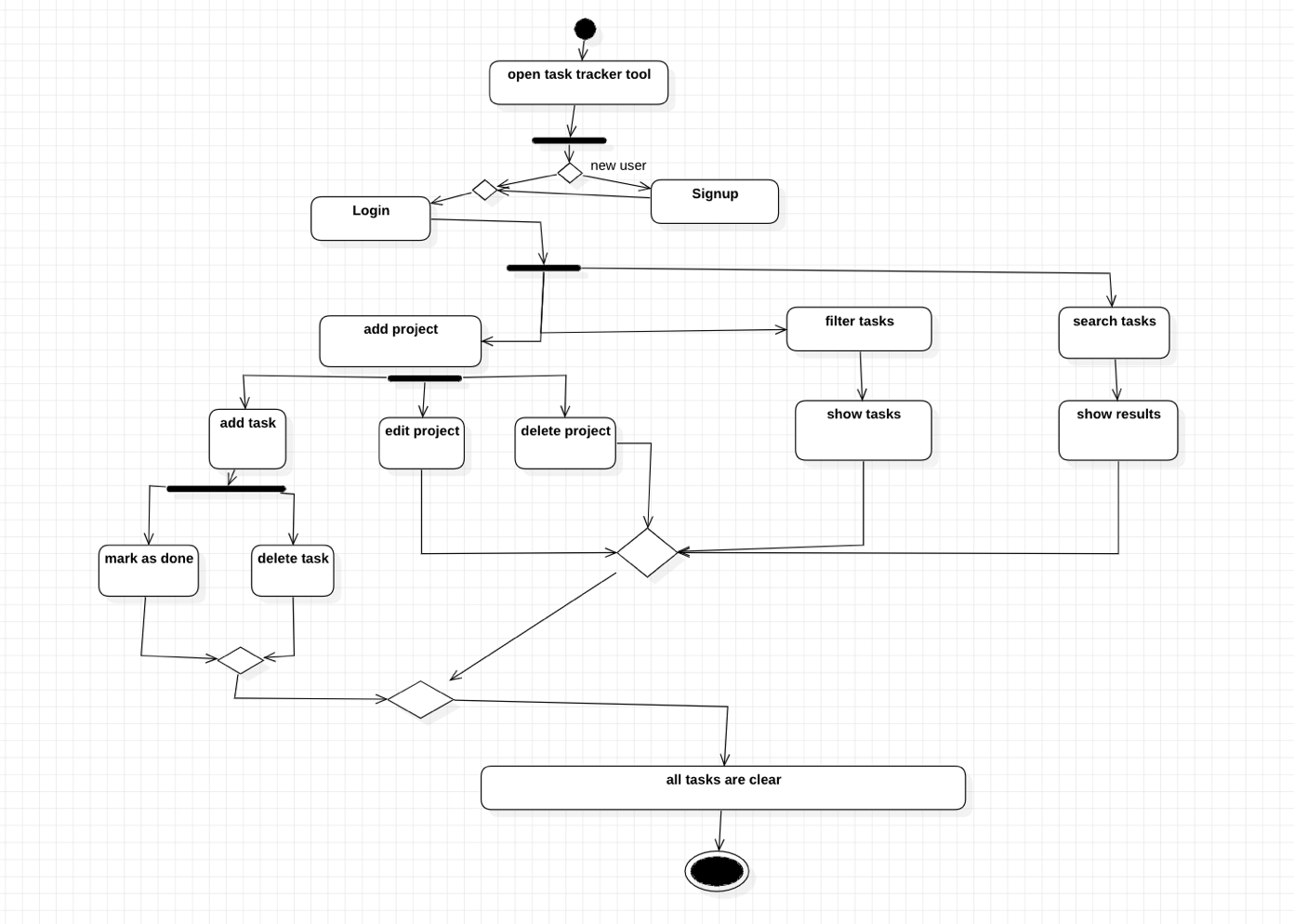
A diagram of a task

Description automatically generated

## **Use Case Diagram:**



## **Activity Diagram:**



## **State Diagram:**

A diagram of a process flow

Description automatically generated

# **Design Patterns**

This section outlines the design patterns used in our application, including Factory Method, Observer, and Strategy patterns. Below are examples of how these patterns are applied:

#### **Factory Method**

* **Definition**: The Factory Method pattern is used to create objects without specifying the exact class of the object that will be created.
* **Integration**: The get\_tokens\_for\_user method in the CustomJWTAuthentication class (located in base/authentication.py) implements the Factory Method pattern by encapsulating the creation of JWT tokens for a user:

**python**

def get\_tokens\_for\_user(user):

refresh = RefreshToken.for\_user(user)

refresh['user\_id'] = user.id

refresh['username'] = user.username

return {

'refresh': str(refresh),

'access': str(refresh.access\_token),

}

This method creates and returns JWT tokens for a user, hiding the object creation logic within a single method, which allows for flexibility in how the tokens are created.

#### **Strategy Pattern**

* **Definition**: The Strategy pattern allows a class to change its behavior based on different conditions or contexts by switching between strategies.
* **Integration**: The TaskFilter component in TaskFilter.jsx uses the Strategy pattern to filter tasks based on their status. The component allows different filtering strategies (e.g., filtering by "done", "in progress", “all”) to be applied dynamically.

#### **Benefits of Using Design Patterns**

* **Separation of Concerns**: Each design pattern focuses on a specific aspect of the system, making the code easier to manage and extend.
* **Flexibility**: The use of design patterns makes the system more adaptable to changes, allowing different behaviors to be swapped in or out without affecting the rest of the system.
* **Scalability**: Design patterns provide structured solutions that can easily grow with the system, facilitating the addition of new features and components without major restructuring.