# Secure Task Manager Web Application Documentation

***W.M. Uditha Erandake Wijekoon – Post Interview Assignment received on 6th March 2025***

Contents

[Secure Task Manager Web Application Documentation 1](#_Toc192505682)

[1. Introduction 1](#_Toc192505683)

[2. Features 2](#_Toc192505684)

[3. Technologies Used 2](#_Toc192505685)

[4. API Endpoints 2](#_Toc192505686)

[Authentication 2](#_Toc192505687)

[Task Management 2](#_Toc192505688)

[5. User Authentication Flow 3](#_Toc192505689)

[6. Challenges Faced 3](#_Toc192505690)

[7. Testing 4](#_Toc192505691)

[7.1 Postman API Testing 4](#_Toc192505692)

[7.2 Web Application Testing 4](#_Toc192505693)

[8. Conclusion 5](#_Toc192505694)

## 1. Introduction

The **Secure Task Manager Web Application** is a web-based task management system that allows users to create, update, and delete tasks efficiently. It includes user authentication, task filtering, managing, and status updates. The backend is developed using Laravel (PHP), while the frontend is built with HTML, Tailwind CSS, and JavaScript. Created two tables for users and tasks as well.

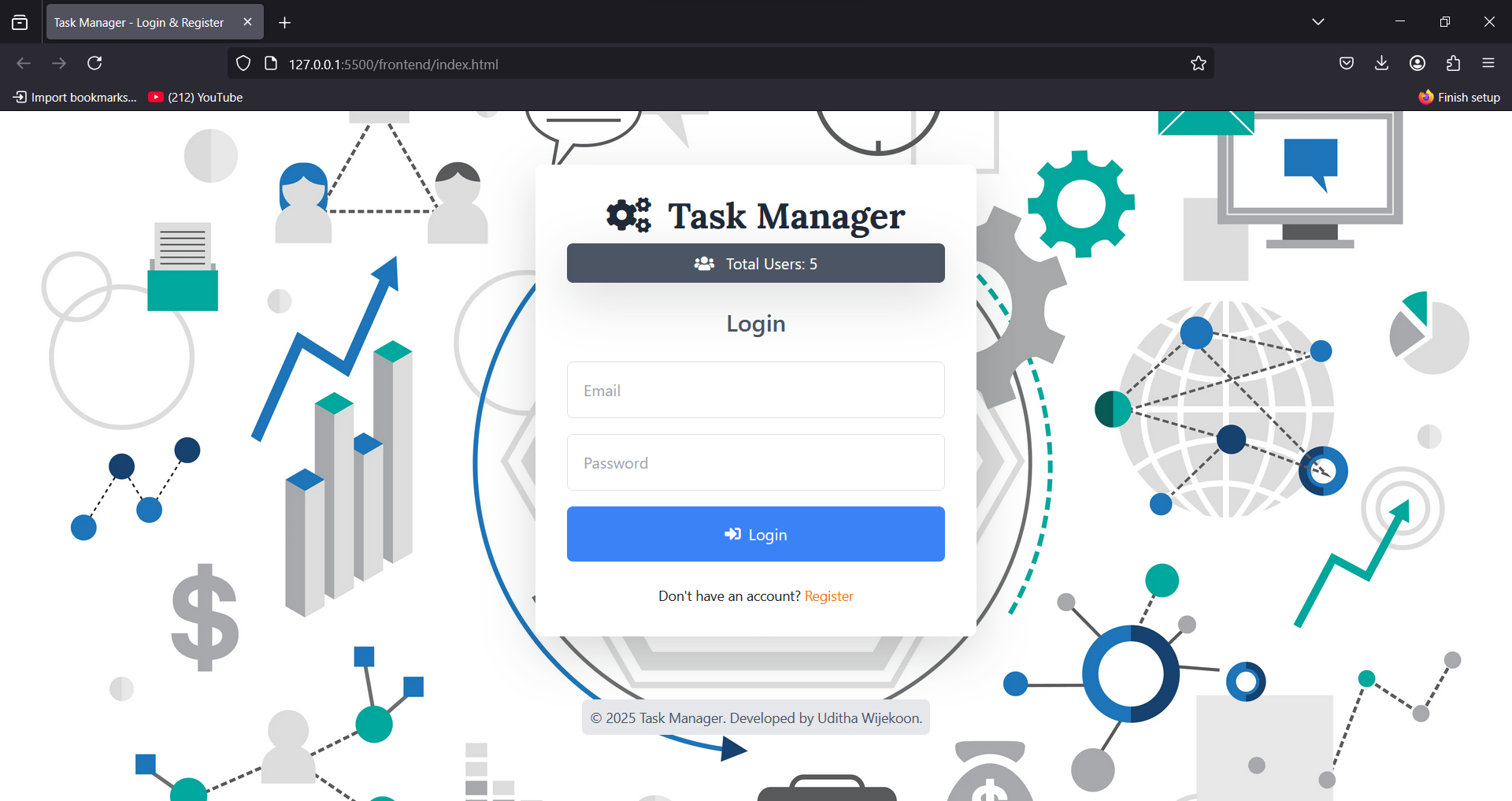


Figure 1 - Task Manager Web Application

## 2. Features

* **User Authentication (JWT)**: Secure login and registration system.
* **Task Management**: CRUD operations for tasks.
* **Task Status Updates**: Tasks can be marked as pending, in progress, or completed.
* **Filter & Sorting**: Tasks can be filtered based on their status.
* **Responsive Design**: Mobile-friendly UI with Tailwind CSS.
* **Security**: Uses JWT for authentication and secure API endpoints.

## 3. Technologies Used

* **Frontend**: HTML, Tailwind CSS, JavaScript, Bootstrap
* **Backend**: Laravel (PHP)
* **Database**: MySQL (used phpMyAdmin for manage the database)
* **Authentication**: JWT (JSON Web Token)

## 4. API Endpoints

### Authentication

* **POST /api/register** - Register a new user (Publicly Accessible)
* **POST /api/login** - Authenticate user & return JWT (Publicly Accessible)
* **POST /api/logout** - Logout user (protected by authentication middleware)
* **GET /api/me** - Get authenticated user details (protected by authentication middleware)
* **GET /api/total**-**users** - get the total number of users (public access)

### Task Management

* **GET /api/tasks** - Fetch all tasks for the authenticated user (Only for logged-in users)
* **POST /api/tasks** - Create a new task (Only for logged-in users)
* **PUT /api/tasks/{id}** - Update a task by ID (Only for logged-in users)
* **DELETE /api/tasks/{id}** - Delete a task by ID (Only for logged-in users)

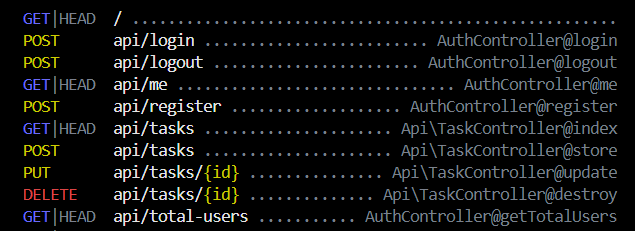


Figure 2 – API Endpoints

## ****5. User Authentication Flow****

1. **Registration**
   * Users register using their user name, email and password.
   * Passwords are hashed for security.
2. **Login**
   * Users enter credentials and receive a JWT token upon successful authentication.
   * Token is stored in local storage.
3. **Logout**
   * Token is removed from local storage, logging out the user.

## 6. Challenges Faced

1. **JWT Authentication Issues**:Initially, setting up JWT authentication in Laravel 12 caused token verification issues. In (*config/auth.php),* the API guard was set to use JWT.

**Solution**: In (*config/auth.php)*, the authentication guard for API requests was updated to use **JWT** and configuring middleware correctly resolved the issue.

1. **Database Relationships**: Maintaining a clear connection between users and their tasks without redundancy was a challenge during early development.

**Solution**: Defined proper foreign key constraints in MySQL and optimized queries for performance.

1. **Button Sizing Consistency**: Ensuring button sizes in task cards required refining CSS styles for smooth appearance.

**Solution**: Used Tailwind CSS utility classes for padding, margins, and transitions.

## 7. Testing

### 7.1 Postman API Testing

#### Authentication Tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case** | **API Endpoint** | **Method** | **Input Data** | **Response** | **Status Code** |
| **Register User** | http://127.0.0.1:8000/api/register | POST | "name": "User Name",  "email": "mail@email.com",  "password": "password" | User created successfully | 201 Created |
| **Login User** | http://127.0.0.1:8000/api/login | POST | "email":"mail@email.com",  "password": "password" | Returns JWT token | 200 OK |
| **Get User Details** | http://127.0.0.1:8000/api/me | GET | JWT Token in Authorization Header | Returns user details | 200 OK |
| **Logout User** | http://127.0.0.1:8000/api/logout | POST | JWT Token in Authorization Header | Logout successful | 200 OK |

#### Task Management Tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case** | **API Endpoint** | **Method** | **Input Data** | **Response** | **Status Code** |
| **Create Task** | http://127.0.0.1:8000/api/tasks | POST | "title": "New Task",  "description": “”,  "status: completed or in-progress or pending" | Task created successfully | 201 Created |
| **Get All Tasks** | http://127.0.0.1:8000/api/tasks | GET | JWT Token in Authorization Header | Returns list of tasks | 200 OK |
| **Update Task** | http://127.0.0.1:8000/api/tasks/(id) | PUT | "Task Updated status" | Task updated successfully | 200 OK |
| **Delete Task** | http://127.0.0.1:8000/api/tasks/(id) | DELETE | Task ID in URL | Task deleted successfully | 200 OK |

### 7.2 Web Application Testing

#### User Authentication Tests

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Steps** | **Result** | **Status** |
| **Register a new user** | Fill out registration form → Submit → Redirect to login | User is successfully registered and redirected to login page | Pass |
| **Login with correct credentials** | Enter valid email & password → Click login | Redirected to dashboard and see a success message | Pass |
| **Login with incorrect credentials** | Enter wrong email/password → Click login | Displays error message: Incorrect email or password | Pass |
| **Register a new user with password length is less than 6 characters** | Fill out registration form with a password that have less than 6 characters → Submit | Displays error message: Password must be at least 6 characters. | Pass |
| **Register a new user with mismatching passwords.** | Fill out registration form with mismatching passwords → Submit | Displays error message: Passwords do not match | Pass |
| **Logout** | Click logout button when logged in | Redirected to login page with success message, session terminated | Pass |

#### Task Functionality Tests

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Steps** | **Result** | **Status** |
| **Add a new task** | Fill in task details → Click "Add Task" | Task appears in the list | Pass |
| **Update task status pending to in-progress** | Click "Move to In Progress" button | Task status updates visually | Pass |
| **Update task status in-progress to completed** | Click " Mark as Completed " button | Task status updates visually | Pass |
| **Delete a task** | Click "Delete" button on a task and confirm | Task is removed from the UI and display a success message | Pass |
| **Add a new task with no description** | Fill in task details with no description → Click "Add Task" | Task appears in the list with “no description” | Pass |
| **Add a new task in progress** | Fill in task details and select In-Progress option→ Click "Add Task" | Task appears in the list with “In-progress” | Pass |

## 8. Conclusion

The Secure Task Manager Web Application provides a streamlined task management system with security-focused implementation, making it a robust solution for users looking to efficiently manage their tasks.

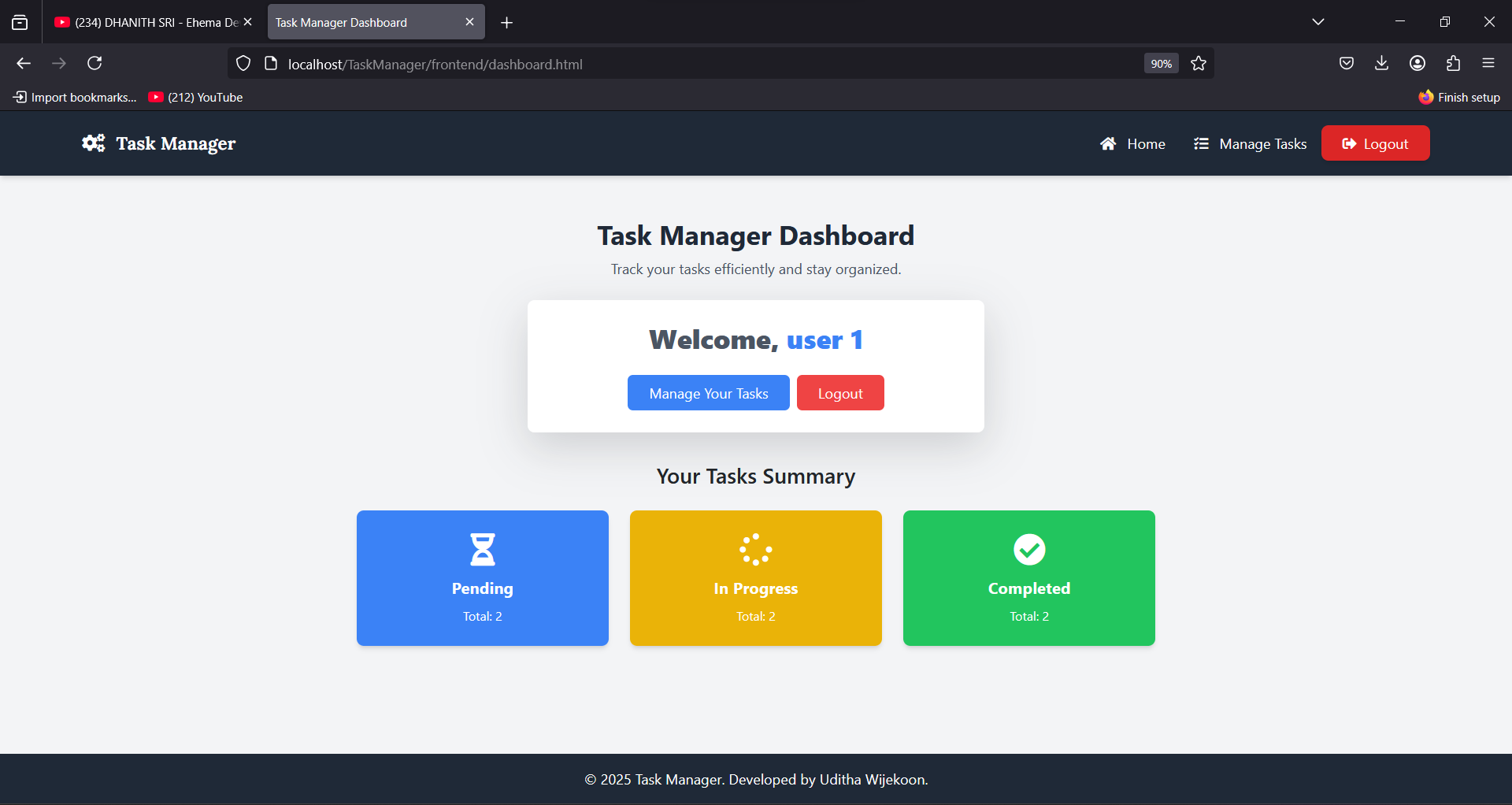


Figure 3 – Task Manager Dashboard