Task Tracking Project

# User Stories

**1. User Registration & Login (Functional)**

**As a new user, I want to register and log in so that I can access and manage my tasks securely.**

**Acceptance Criteria:**

* Registration form requires name, email, and password.
* Password is encrypted in the database.
* JWT token is issued on successful login.

**2. Task Creation (Functional)**

**As a logged-in user, I want to create a task with a title, description, estimate time.**

**Acceptance Criteria:**

* All fields (title, status, estimate) are required.
* Task is saved in the PostgreSQL database.
* Tasks appear in the dashboard after creation.

**3. Task Viewing (Functional)**

**As a user, I want to view a list of my tasks with their current status and progress so that I can track what needs to be done.**

**Acceptance Criteria:**

* Tasks are retrieved from the backend.
* Tasks are listed with title, status, and progress bar.

**4. Task Editing (Functional)**

**As a user, I want to edit an existing task so that I can update its information when things change.**

**Acceptance Criteria:**

* Form pre-fills with existing task data.
* After editing, data is updated in the DB and UI.

**5. Task Deletion (Functional)**

**As a user, I want to delete tasks I no longer need so that I can keep my dashboard clean.**

**Acceptance Criteria:**

* Clicking delete removes the task from DB.

**6. Time Logging (Functional)**

**As a user, I want to log the number of hours I spend on a task so that I can track my effort.**

**Acceptance Criteria:**

* Time log input accepts hours.
* Progress bar updates based on estimate vs. actual.

**7. Change Task Status (Functional)**

**As a user, I want to change a task’s status (To-Do, In Progress, Done) so that I can reflect its current state.**

**Acceptance Criteria:**

* Status dropdown is available in the task card/form.

**8. Logout Functionality (Functional)**

**As a user, I want to log out so that I can securely end my session.**

**Acceptance Criteria:**

* JWT token is removed from local storage.

# API Definitions

**POST /api/auth/register**

* **Description**: Register a new user.
* **Request Body**:

{

"username": "ahmed",

"email": "ahmed@example.com",

"password": "securePassword123"

}

* **Response (201)**:{

"message": "User registered successfully"

}

**POST /api/auth/login**

* **Description**: Authenticate user and return JWT.
* **Request Body**:

{

"email": "ahmed@example.com",

"password": "securePassword123"

}

* **Response (200)**:

{

"token": "eyJhbGciOiJIUzI1NiIsInR..."

}

**GET /api/tasks**

* **Description**: Get all tasks for the logged-in user.
* **Headers**: Authorization: Bearer <JWT>
* **Response (200)**:

[

{

"id": 1,

"title": "Finish API",

"description": "Write the API contract section",

"status": "In Progress",

"estimate": 4,

"logged\_time": 2

}

]

**POST /api/tasks**

* **Description**: Create a new task.
* **Headers**: Authorization: Bearer <JWT>
* **Request Body**:

{

"title": "New Task",

"description": "Details about the task",

"status": "To-do",

"estimate": 5

}

* **Response (201)**:

{

"message": "Task created successfully",

"task\_id": 2

}