🟧 🟪 🟦 // to use as priorities

**🟦 1. User Signup (Functional)**

**Story:**  
As a new user, I want to create an account using an email and password so that I can access the application and manage my tasks.

**Acceptance Criteria:**

* Input validation for email & password
* API endpoint creates a new user
* JWT is returned upon successful signup
* User is redirected to an empty dashboard or maybe a tutorial

**🟦 2. User Login (Functional)**

**Story:**  
As an existing user, I want to log in securely so that I can access my personal task board.

**Acceptance Criteria:**

* Valid credentials return JWT
* JWT stored locally
* Invalid login shows error message

### 🟦 3. ****Create Task**** (Functional)

**Story:**  
As a logged in user, I want to create a new task with a title, description, status, and estimated time so that I can plan my work.

**Acceptance Criteria:**

* User presses a “New Task” button that opens the task creation window
* Task form accepts required fields
* New task appears in task list immediately
* Data saved to DB using API

### 🟦 4. ****Update Task Status**** (Functional)

**Story:**  
As a logged in user, I want to change a task’s status so that I can track its progress through my workflow.

**Acceptance Criteria:**

* Status dropdown shows "To-do", "In Progress", and "Done"
* Selecting a new status updates DB immediately

### 🟦 5. Edit Task (Functional)

**Story:**  
As a logged-in user, I want to edit a task’s title, description, or estimated time so that I can update my plans.

**Acceptance Criteria:**  
• User clicks an “Edit” button to open a pre-filled form  
• Updated fields are saved  
• Task updates instantly in the UI without page refresh  
• Invalid inputs (e.g., empty title) show validation error messages

### 🟧 6. ****Log Time on Task**** (Functional)

**Story:**  
As a logged in user, I want to log hours spent on a task so that I can track how much time I’ve invested.

**Acceptance Criteria:**

* Time input accepts positive numbers
* Time is added cumulatively to the task
* Progress bar updates based on logged vs. estimated time

### 🟦 7. ****Delete Task**** (Functional)

**Story:**  
As a logged in user, I want to delete tasks I no longer need so that I can keep my board clean.

**Acceptance Criteria:**

* Clicking delete prompts confirmation
* Task is removed from UI
* Backend sets task as Done (if finished) or removes it from the DB (if not finished)

### 🟧 8. ****Logout**** (Functional)

**Story:**  
As a logged in user, I want to securely log out so that no one else can access my account from my device.

**Acceptance Criteria:**

* Logout button clears local storage JWT
* User is redirected to login page

### 🟪 9. ****Stats & Productivity Insights Tab**** (Functional)

**Story:**  
As a logged in user, I want to view stats about my completed tasks so that I can track my productivity and stay motivated.

**Acceptance Criteria:**  
• Shows number of tasks completed today, this week, this month, and this year  
• Shows average tasks completed per week  
• Shows current productivity streak (e.g., “4 days in a row”)  
• Shows completion rate for the current month (e.g., “12/20 tasks done”)  
• Highlights most productive day of the week (e.g., “Bro you crushed Wednesday with finishing 3 tasks!”)  
• Displays badges for milestones (e.g., “7-Day Streak”, “100 Tasks Completed”)  
• Stats are user-specific and protected by JWT  
• UI updates automatically when tasks are completed

**NON-FUNCTIONAL FEATURES**

**🟧 1. Performance Optimization (Non-Functional)**

**Story:**  
As a user, I want the app to load quickly so that I can manage tasks efficiently.

**Acceptance Criteria:**  
• All pages (login, dashboard, stats) load within 2 seconds during testing  
• API responses return in under 500ms for typical requests

**🟦 2. Application Security (Non-Functional)**

**Story:**  
As a user, I want my data to be protected so that unauthorized users cannot access my tasks.

**Acceptance Criteria:**  
• All task-related API routes require a valid JWT in the Authorization header  
• Input validation prevents SQL injection  
• Passwords are hashed before storage