2 Test plan

2.1 Introduction

The objective of this test plan is to define the strategy, scope, resources, and schedule for the testing activities of the **Planora - Online Project Management System**. The primary goal is to verify that the software meets the requirements specified in the SRS document, functions correctly across different modules, and provides a stable user experience before the final release.

2.2 Scope of Testing

The testing process will cover the core functionalities of the Planora system, focusing on the following areas:

In-Scope:

- **User Management:** Authentication (Login), Profile Management, and Role-based Access Control (Manager, Member, Viewer).
- **Project Management:** CRUD operations for Projects, Member assignment, and Project settings.
- **Task Management (Kanban Board):** Creating tasks, Drag-and-drop status updates, filtering, and editing task details.
- **API Integration:** Verifying RESTful API endpoints for data consistency and security.
- **User Interface (UI):** Responsiveness, Theme toggling (Dark/Light mode), and visual consistency.
- **Performance:** Basic load time verification for critical dashboards.

Out-of-Scope:

- **Stress/Load Testing:** High-volume concurrent user testing (beyond the scope of this course project).
- **Third-party Payment Gateway Integration:** Only mock testing for subscription upgrades (if applicable), not live financial transactions.

2.3 Test Strategy

The team will adopt a combination of **Manual Testing** and **API Testing** approaches:

1. **Functional Testing:**
    - Focus: Validate that features work according to the user stories and use cases.
    - Method: Manual execution of test cases via the ReactJS frontend.
    - Key Areas: Task CRUD, Kanban board interactions, Filter logic.
2. **API Testing:**
    - Focus: Ensure backend logic, data validation, and HTTP status codes are correct before UI integration.
    - Tool: Postman.
    - Key Areas: User authentication flows, Role permission checks (RBAC), Data retrieval.
3. **UI/UX Testing:**
    - Focus: Visual appearance, layout stability, and usability.
    - Method: Visual inspection on Google Chrome and Microsoft Edge.
    - Key Areas: Dark mode toggle, Modal overlays, Drag-and-drop smoothness.
4. **Performance Testing:**
    - Focus: Page load speed and API response latency.
    - Tool: Chrome DevTools (Network Tab).
    - Key Metric: Dashboard load time < 3 seconds.

2.4 Test Environment

To ensure consistent results, testing will be conducted in the following environment:

- **Hardware:** Personal Laptops (Windows 10/11).
- **Browser:** Google Chrome and Microsoft Edge.
- **Frontend:** ReactJS application running on localhost:5001
- **Backend:** Node.js/Express server running on localhost:5173
- **Database:** MongoDB Atlas (Cloud).
- **Network:** Stable Wifi/4G connection.

2.5 Test Tools

The following tools are utilized to support the testing process:

- **Test Management:** Excel/Word (for maintaining Test Case documents).
- **API Testing:** Postman (for executing TC_API_001 to TC_API_005).
- **Performance Analysis:** Chrome Developer Tools (Lighthouse & Network tab).
- **Bug Tracking:** Github Issues or Trello/Planora Internal Board.

2.6 Defect Management

Defects found during testing will be categorized and tracked using the following severity levels:

- **Critical:** System crash, data loss, or inability to login. (Must fix immediately).
- **High:** Major functionality broken (e.g., Cannot create a task, Drag-and-drop fails).
- **Medium:** Minor logic errors or UI issues that do not block usage.
- **Low:** Cosmetic issues (typos, alignment) or suggestions.