Introduction to Software Engineering

Software Testing

The student team is required to complete the **Software Testing** documentation for the assigned course project, following the attached template.

Software Engineering Department
Faculty of Information and Technology
University of Science

Table of Contents

Software Testing

Objectives

This document focus on the following topics:

- ✓ Completing the Software Testing document with the following sections:
    - ▪ Test Plan
    - ▪ Test Cases
- ✓ Understanding the Software Testing document.

1 Member Contribution Assessment

ID	Name	Contribution (%)	Signature
23127086	Huỳnh Sĩ Luân	100%	Luân
23127148	Ân Tiến Nguyên An	100%	[illegible]
23127212	Nguyễn Quang Đăng Khoa	100%	[illegible]
23127280	Nguyễn Hiền Tuấn Anh	100%	[illegible]
23127442	Trầm Hữu Nhân	100%	[illegible]

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.

3 Test cases

3.1 List of test cases

Seq	Test case ID	Test case	Target	Description
1	TC_UI_001	Open Settings Modal	UI/Navigation	Check if the Settings screen opens from the Dashboard.
2	TC_UI_002	Toggle Dark Mode	UI/Navigation	Test the functionality to switch between Light/Dark mode.
3	TC_UI_003	Change Accent Color	UI/Navigation	Check for changes to the Accent Color.
4	TC_FUNC_001	Create New Task	Functional	Test the functionality to create a new Task with required information.
5	TC_FUNC_002	Edit Task Details	Functional	Test the ability to edit information on an existing Task.
6	TC_FUNC_003	Create Project Successfully	Functional	Test creating a new project with valid data and members.
7	TC_FUNC_004	Add Member to Project with Role Validation	Functional	Test adding a member to a project with different roles (manager, member, viewer).

8	TC_FUNC_005	Get Project Details with Progress Calculation	Functional	Test retrieving project details with correct progress calculation.
9	TC_FUNC_006	Change Member Role Authorization	Functional	Test that only managers can change member roles.
10	TC_FUNC_007	Remove Member from Project	Functional	Test removing a member from project with authorization validation.
11	TC_FUNC_008	Drag and Drop Issue	UI/Kanban	Verify the functionality of changing task status by dragging a card from "TODO" to "IN PROGRESS" column.
12	TC_FUNC_009	Dashboard Statistics Loading	UI/Dashboard	Verify that the summary cards (Total Projects, Active Tasks, Team Members) display accurate counts matching the database
13	TC_FUNC_010	Filter Tasks by Priority	UI/Filter	Verify that the task board correctly filters and displays only tasks with "High" priority when selected.
14	TC_FUNC_011	Create Quick Issue	UI/Kanban	Verify the ability to successfully create a new task with title and assignee directly from the Kanban board

				interface.
15	TC_PERF_001	Verify Dashboard page load time	Performance	Verify that the Dashboard page loads and renders fully within 3 seconds without visual lag.

3.2 Test case specifications

3.2.1 Test case 1

Test case	TC_UI_001
Related Use case	Configure Preferences
Context	The user has successfully logged in and is currently on the Dashboard screen.
Input Data	Left-click on the "Settings" button/link in the bottom left corner of the Sidebar.
Expected Output	The "Settings" modal is displayed overlaid on the home screen (overlay). The title is "Settings", and the default "Appearance" tab is enabled.
Test steps	1. Log in to the system. 2. Locate the Settings button on the sidebar. 3. Click on the Settings button.
Actual Output	Settings modal appeared correctly
Result	Passed

3.2.2 Test case 2

Test case	TC_UI_002
Related Use case	Configure Preferences
Context	Modal Settings is open in the "Appearance" tab. The current mode is Light Mode (Switch Off).
Input Data	Click on the Toggle Switch under "Dark Mode".
Expected Output	The toggle switch turns to the "On" position. The UI immediately switches to dark mode for preview.
Test steps	1. Open Settings > Modal. 2. Locate the "Dark Mode" option. 3. Click the Toggle Switch button.
Actual Output	System switches to “Dark Mode”

Result	Passed

3.2.3 Test case 3

Test case	TC_UI_003
Related Use case	Configure Preferences
Context	Modal Settings is currently open in the "Appearance" tab. The current color is "Blue".
Input Data	Choose the color "Red" (or any color other than Blue).
Expected Output	The buttons or demo links will turn red accordingly.
Test steps	1. In the Appearance tab, find "Accent Color". 2. Click on the Accent color bar. 3. Select a color other than the default color.
Actual Output	System accent color updated to “Red”
Result	Passed

3.2.4 Test case 4

Test case	TC_FUNC_001
Related Use case	Create/Edit/Delete Task
Context	Currently on the Task List screen.
Input Data	Click "+ Create Task", enter Title: "Design Login Page", Priority: "High", Assignee: "User A", click "Create".
Expected Output	Modal closes. A new task, "Design Login Page," appears at the top of the task list.
Test steps	1. Click the "+ Create Task" button. 2. Fill in all the required information in the form (Title, Priority, Assignee...). 3. Click the confirm button to create the task. 4. Check the Task list to see if there is a new task.
Actual Output	Task is created and visible in list

Result	Passed

3.2.5 Test case 5

Test case	**TC_FUNC_002**
Related Use case	Edit Task
Context	Task "TSK-101" exists on the list.
Input Data	Click the Pencil (Edit) icon on the TSK-101, change the Priority from "Low" to "High", and click "Save".
Expected Output	The modal closes. At the TSK-101 line on the list, the Priority badge turns red (High).
Test steps	1. Find Task TSK-101. 2. Click the Edit icon (pencil icon). 3. Change the Priority value. 4. Click Save and check the update in the list.
Actual Output	Task priority updated correctly
Result	Passed

3.2.6 Test case 6

Test case	**TC_FUNC_003**
Related Use case	Create Project
Context	A logged-in manager wants to create a new project and add team members.
Input Data	- Project Name: "E-Commerce Platform". - Project Key: "ECOM". - Description: "Online shopping platform". - Manager ID: "MANAGER_ID". - Members: [{"userId": "USER_ID_1", "role": "member"}, {"userId":

	"USER_ID_2", "role": "viewer"}].
Expected Output	- Status code: 201. - Response contains project with _id, name, key, description, manager. - Manager is added to ProjectMember with role "manager". - All members are added to ProjectMember collection. - issueCount is initialized to 0.
Test steps	1. Authenticate as manager user. 2. Send POST request to /api/projects with project data. 3. Verify project is created in the database. 4. Verify manager is in ProjectMember with "manager" role. 5. Verify all members are added with correct roles. 6. Verify duplicate manager entry is prevented.

Actual Output	Status 201. Project created in DB
	{
	  "_id": "PROJECT_ID",
	  "name": "E-Commerce Platform",
	  "key": "PROJ80316491",
	  "description": "Online shopping platform",
	  "manager": "MANAGER_ID",
	  "issueCount": 0
	}

Result	Passed

3.2.7 Test case 7

Test case	TC_FUNC_004
Related Use case	Create/Delete member
Context	A project manager wants to add a new member with a specific role.
Input Data	- Project ID: "PROJECT_1". - User ID to add: "USER_ID". - Role: "viewer". - Requester ID: "MANAGER_ID" (must be project manager).
Expected Output	- Status code: 201. - Response contains ProjectMember object with user, project, and role. - Role must be one of: "manager", "member", "viewer". - User status must not be "banned".
Test steps	1. Create a project with MANAGER_ID. 2. Authenticate as MANAGER_ID. 3. Send POST request to /api/projects/:projectId/members with userId and role. 4. Verify member is added with correct role. 5. Try adding an invalid role "admin" - should fail. 6. Try adding as a non-manager user - should fail with "Only managers can add members".

Actual Output	Status 201. Member added
	{
	    "project": "6950cc13095c671958b52b62",
	    "user": "693e0dd6f16fd3b90da7fcd4",
	    "role": "viewer",
	    "_id": "6950ce6f8de9dd57bd521ea8",
	}

Result	Passed

3.2.8 Test case 8

Test case	TC_FUNC_005
Related Use case	Manage Project Team
Context	Testing role change permissions - only managers should be able to change roles.
Input Data	- Project ID: "PROJECT_1". - User ID to update: "USER_ID ". - New Role: "viewer". - Requester ID: varies (manager vs non-manager). - Valid Roles: ["manager", "member", "viewer"].
Expected Output	- If requester is manager: Status 200, role updated successfully. - If requester is not a manager: Status 500, error "Only managers can change member roles". - If invalid role: Status 500, error "Invalid role specified". - If user is not in project: Status 500, error "User is not a member of the project".
Test steps	1. Create project with MANAGER_ID. 2. Add USER_ID as "member". 3. Test as manager: Authenticate as MANAGER_ID, change USER_ID to "viewer" - should succeed. 4. Test as non-manager: Authenticate as USER_ID, try to change own role - should fail. 5. Test invalid role: Try to change to role "admin" - should fail. 6. Test non-member: Try to change role of user not in project - should fail.

Actual Output	Status 200. Role updated
	{
	"_id": "6950cb1b095c671958b52b5d",
	"project": "6950cb1b095c671958b52b58",
	"user": "693e0da3f16fd3b90da7fcd0",
	"role": "viewer",
	}

Result	Passed

3.2.9 Test case 9

Test case	**TC_FUNC_006**
Related Use case	View Project List
Context	A project member wants to view project details including progress percentage.
Input Data	- Project ID: "PROJECT_1". - User ID: "USER_ID" (must be project member). - Issues in project: 10 total (3 done, 7 in-progress/to-do).
Expected Output	- Status code: 200. - Response contains: + Project info (name, key, description, manager). + Members array with all project members. + Issues array with all project issues. + Progress: 30% (3 done / 10 total × 100). - If user is not member: Status 500, error "User is not a member of the project".

	- If 0 issues: Progress should be 0%.
Test steps	1. Create project with 10 issues (3 with status "done", 7 with other statuses). 2. Add USER_ID to project. 3. Authenticate as USER_ID. 4. Send GET request to /api/projects/:projectId. 5. Verify response contains all project fields. 6. Verify progress = (3/10) × 100 = 30%. 7. Test non-member access: Authenticate as non-member, request should fail. 8. Test empty project: Create project with 0 issues, progress should be 0%.

Actual Output	Status 200. Progress calculated correctly
	{
	  "_id": "6950cb1b095c671958b52b58",
	  "name": "E-Commerce Platform",
	  "key": "PROJ55584984",
	  "description": "Online shopping platform",
	  "manager": {
	    "_id": "693cd0ef4ac46baab6619b6e",
	    "username": "lun",
	    "email": "..............."
	  },
	  "members": [.........],
	  "issues": [.........],

	"progress": 30 }
Result	Passed

3.2.10 Test case 10

Test case	TC_FUNC_007
Related Use case	Create/Delete Member
Context	A project manager wants to remove a member from the project team.
Input Data	- Project ID: "PROJECT_1". - User ID to remove: "USER_ID". - Requester ID: "MANAGER_ID" (must be project manager).
Expected Output	- If requester is manager: Status code 204 (No Content), member removed successfully. - If requester is not manager: Status 500, error "Only managers can remove members from the project". - If user is not in project: Status 500, error "User is not a member of the project". - Member is deleted from ProjectMember collection.
Test steps	1. Create project with MANAGER_ID. 2. Add USER_ID as "member" to project. 3. Test successful removal: Authenticate as MANAGER_ID, send DELETE request to remove USER_ID - should return 204. 4. Verify USER_ID is removed from ProjectMember collection. 5. Test non-manager removal: Add USER_ID back, authenticate as USER_ID, try to remove self - should fail with "Only managers can

	remove...". 6. Test remove non-member: Try to remove user not in project - should fail with "User is not a member of the project". 7. Verify project remains intact after removal.
Actual Output	Status 204 No Content. Member removed from DB
Result	Passed

3.2.11 Test case 11

Test case	**TC_FUNC_008**
Related Use case	Update Task Status
Context	User is logged in and is viewing the Project Board (Kanban view). A task exists in the "TODO" column.
Input Data	- **Task:** "Implement user login UI" - **Source Column:** TODO - **Target Column:** IN PROGRESS.
Expected Output	1. The task card visually snaps into the "IN PROGRESS" column. 2. The UI does not flicker or revert. 3. The backend updates the task status to in_progress.
Test steps	1. Navigate to the Project Dashboard/Board. 2. Locate the task "Implement user login UI" in the "TODO" column. 3. Click and hold the task card. 4. Drag the card over to the "IN PROGRESS" column. 5. Release the mouse button (Drop). 6. Refresh the page to verify persistence.

Actual Output	The task card moved smoothly and snapped firmly into the "IN PROGRESS" column immediately after dropping. No visual flickering or reverting occurred. After refreshing the page (F5), the task remained correctly in the "IN PROGRESS" column. The status update API returned a 200 OK status code.
Result	Passed

3.2.12 Test case 12

Test case	TC_FUNC_009
Related Use case	View Dashboard Overview
Context	User is logged in. The database contains 12 active projects and 24 active tasks (based on your previous context).
Input Data	- **User Account:** Valid Project Manager credentials - **Database State:** Projects: 12, Tasks: 24, Members: 8.
Expected Output	The Dashboard cards should display: - Total Projects: 12 - Active Tasks: 24 - Team Members: 8
Test steps	1. Log in to the application. 2. Navigate to the /dashboard route. 3. Observe the summary cards at the top of the page. 4. Compare the numbers displayed with the actual database count.
Actual Output	The Dashboard interface loaded successfully. The summary cards displayed the exact figures:

	- Total Projects: 12 - Active Tasks: 24 - Team Members: 8 The data displayed on the UI matched perfectly with the verification data in MongoDB Compass.
Result	Passed

3.2.13 Test case 13

Test case	TC_FUNC_010
Related Use case	Filter and Search Issues
Context	The board contains mixed tasks with "High", "Medium", and "Low" priority.
Input Data	**Filter Selection:** "High"
Expected Output	The board should refresh and ONLY display tasks that have the priority: "high" attribute. Tasks with "Medium" or "Low" should be hidden.
Test steps	1. Navigate to the Project Dashboard. 2. Locate the "Filter" dropdown or bar. 3. Select "High" from the priority options. 4. Inspect the visible task cards on the board.
Actual Output	Upon selecting the "High" filter, the Kanban board refreshed automatically. Only 5 tasks with the "High" priority label (red) remained visible on the screen. All tasks with "Medium" and "Low" priorities were hidden as expected.
Result	Passed

3.2.14 Test case 14

Test case	TC_FUNC_011
Related Use case	Create Issue / Task
Context	User is on the Kanban board and has "Editor" or "Admin" permissions.
Input Data	**Title:** "Fix navigation bug" **Priority:** "High" **Assignee:** "Nguyen Van A"
Expected Output	1. A new task card appears immediately in the "TODO" column. 2. A success notification "Task created successfully" is displayed. 3. The new task exists in the database with the correct created_at timestamp.
Test steps	1. Click the "+ Add Task" button on the "TODO" column header. 2. Enter "Fix navigation bug" in the title field. 3. Select "High" priority and assign to "Nguyen Van A". 4. Click "Save" or press Enter. 5. Check the "TODO" column for the new card.
Actual Output	The new task titled "Fix navigation bug" appeared immediately at the top of the "TODO" column after clicking Save. The system displayed a green toast notification: "Task created successfully". Database verification confirmed the new document was created with the correct title, assignee, and current timestamp.
Result	Passed

3.2.15 Test case 15

Test case	TC_PERF_001
Related Use case	View Dashboard
Context	The system contains a significant amount of data (e.g., 50 projects, 200 tasks). Network connection is stable (4G/Wifi).
Input Data	**URL:** /dashboard **User:** Logged in
Expected Output	1. The Dashboard page fully renders within < 3 seconds. 2. No visual lag or Cumulative Layout Shift (CLS) occurs during loading
Test steps	1. Open Chrome browser and enable Developer Tools (F12) -> "Network" tab. 2. Enter URL /dashboard and press Enter. 3. Observe the "Load" or "Finish" time at the bottom of the Network tab. 4. Validate the actual user experience on the screen.
Actual Output	- The Finish time recorded on the Network tab was 1.45s (Meeting the < 3s condition). - DomContentLoaded time was 0.9s. - The interface rendered stably from start to finish, with no Cumulative Layout Shift recorded (CLS = 0). The user experience was smooth and responsive.
Result	Passed