Application Under Test (AUT):
“TaskFlow” - Task Management Web Application

1. Introduction

This document describes the **Software Test Plan** for ***TaskFlow***, a task management web application designed to help individuals and teams organize, prioritize, and track their daily work efficiently. The plan outlines the testing strategy, objectives, scope, resources, and schedule required to ensure the application’s quality before release.

TaskFlow provides a centralized platform for **task creation, assignment, scheduling, progress tracking, and collaboration** within teams. It aims to enhance productivity by offering features such as **real-time updates, notifications, role-based access, deadline reminders, and performance analytics.**

The purpose of this test plan is to define a systematic approach for verifying the functionality, performance, usability, and reliability of the TaskFlow application before its release. It outlines the **testing objectives, scope, test strategy, environment setup, resource requirements, risk analysis, and schedule.**

This document serves as a **guideline for the QA team** to plan, design, execute, and monitor all testing activities throughout the project lifecycle, ensuring that TaskFlow meets the **expected quality standards**, aligns with **business requirements**, and provides an **error-free experience** to end-users across different browsers and devices.

2. Objectives

The primary objective of this Test Plan is to ensure that the **TaskFlow** application meets its functional and non-functional requirements through systematic and well-defined testing activities. The goals of the testing process include:

1. **Verification of Core Functionalities:**
   Ensure that all primary features — including task creation, editing, deletion, assignment, and progress tracking — function correctly under various user scenarios and data conditions.
2. **Reliability and Stability:**
   Validate that the TaskFlow system remains stable and reliable during continuous use, concurrent user sessions, and unexpected input cases without performance degradation or crashes.
3. **Usability and Accessibility:**
   Assess the user interface for intuitiveness, clarity, and accessibility. Confirm that the application provides a seamless and user-friendly experience for both technical and non-technical users.

4. **Performance Evaluation:**
   Measure system performance in terms of response time, task loading speed, and server efficiency under normal and peak load conditions.
5. **Early Bug Detection and Coverage:**
   Identify defects as early as possible in the development lifecycle to reduce rework costs and ensure complete functional, integration, and regression test coverage.
6. **Integration Validation:**
   Verify proper integration with external services and APIs, including user authentication (Google OAuth) and notification systems (email and in-app alerts).
7. **Cross-Platform and Cross-Browser Compatibility:**
   Ensure consistent performance and layout across supported browsers (Chrome, Firefox, Edge) and devices (desktop, tablet, mobile).
8. **Quality Assurance Alignment:**
   Confirm that all deliverables align with TaskFlow's quality assurance standards and organizational acceptance criteria before production deployment.

3. Scope

The scope of this Test Plan defines the features, functionalities, and components of the **TaskFlow** web application that will be tested (in-scope), as well as those that are excluded from the current testing cycle (out-of-scope).
This ensures clarity on the testing boundaries and helps the QA team focus efforts effectively.

3.1 In-Scope Testing

The following components and features will be included in the current testing cycle:

1. **User Registration and Authentication:**
   - Verification of login and signup functionalities using both Email and Google OAuth.
   - Validation of password reset, account verification, and session management processes.
2. **Dashboard and Task Board Functionality:**
   - Testing of user-specific dashboards displaying active, pending, and completed tasks.
   - Validation of task categorization by project, due date, and priority.
3. **Task Creation, Editing, and Deletion:**
   - Functional testing to ensure users can create new tasks, modify existing ones, and delete them successfully.
   - Verification that updates reflect immediately across all connected users and sessions.
4. **Task Assignment and Deadline Alerts:**
   - Testing of task delegation features among team members.
   - Validation of deadline alerts, reminders, and overdue notifications.

5. **Activity Log and Real-Time Notifications:**
    - Verification that all user actions (task updates, status changes, assignments) are captured in the activity log.
    - Testing real-time notifications for new assignments, updates, and completed tasks using WebSocket or Firebase services.
6. **Cross-Browser and Compatibility Testing:**
    - Testing on Chrome, Firefox, and Edge browsers.
    - Ensuring consistent layout, performance, and behavior across different screen sizes (desktop, tablet, mobile web).
7. **Security and Data Validation:**
    - Testing input validation, session timeout, and protection against unauthorized access (SQL injection, XSS).

3.2 Out-of-Scope

The following components and functionalities are **excluded** from this testing cycle due to pending development or future release plans:

1. **Admin Analytics Dashboard:**
    - Features related to administrative data visualization and team performance analytics are not part of this release.
2. **Mobile Application Version:**
    - Testing of native Android or iOS versions of TaskFlow is excluded, as this test plan focuses solely on the web platform.
3. **Integration with Third-Party Calendar APIs:**
    - External calendar synchronization (Google Calendar, Outlook) is under development and will be included in future testing phases.
4. **Performance Benchmarking Beyond Planned Limits:**
    - Extreme load testing beyond 1,000 concurrent users is deferred to a separate performance testing plan.

4. Test Items

Module	Description	Expected Output
Login & Signup	Verify authentication with email and Google	Successful login and session
Dashboard	Display all active tasks for the user	Accurate task listing
Task CRUD	Create, read, update, delete tasks	Correct data update in DB
Task Assignment	Assign tasks to team members	Notification sent to assignee

Notifications	Trigger on task updates and deadlines	Real-time notification delivered

5. Features Not to Be Tested

- Database optimization queries.
- System metrics collection for future analytics.
- Admin role permissions (feature not released).

6. Test Environment

Testing will be conducted under simulated production conditions.

Parameter	Configuration
Server Environment	Node.js backend on AWS EC2
Database	MongoDB Atlas
Client Browser	Chrome, Firefox, Edge
Testing Tools	Selenium, Postman, JMeter, Jest
Network	10 Mbps broadband and 4G network tests

7. Test Strategy

The testing strategy for TaskFlow adopts a hybrid approach, combining manual and automated testing to ensure comprehensive coverage of both functional and non-functional requirements.

The goal is to identify defects efficiently, verify system integrity, and ensure the product meets the business and technical specifications before deployment.

7.1 Testing Approach

1. Unit Testing

- **Responsibility:** Development Team
- **Tools Used:** Jest (for backend), React Testing Library (for frontend)
- **Objective:** Validate individual code components, such as functions, classes, and methods, to ensure they operate as intended in isolation.
- **Examples:**
    - Verify `addTask()`, `updateTask()`, and `deleteTask()` functions.
    - Check backend API endpoint responses and error handling.

- **Expected Outcome:** All modules pass unit-level test cases with 100% functional coverage.

2. Integration Testing

- **Responsibility:** QA Team
- **Tools Used:** Postman, Newman, MongoDB Compass
- **Objective:** Ensure that modules and APIs interact correctly between frontend, backend, and database layers.
- **Focus Areas:**
    - Data flow validation from UI → API → Database → UI.
    - Authentication and notification API integration tests.
- **Expected Outcome:** All modules communicate without data loss, API errors, or logical mismatches.

3. System Testing

- **Responsibility:** QA Team
- **Tools Used:** Selenium WebDriver, JMeter
- **Objective:** Validate the complete end-to-end workflow in a **staging environment** using real-world test data.
- **Focus Areas:**
    - Task creation, modification, and deletion workflow.
    - Task assignment notifications and activity logs.
    - Cross-browser and UI consistency.
- **Expected Outcome:** The application performs as expected across all modules without functional or UI defects.

4. User Acceptance Testing (UAT)

- **Responsibility:** QA Team and End Users (Client Representatives)
- **Tools Used:** Google Docs (for feedback), Jira (for tracking UAT issues)

- **Objective:** Validate that the TaskFlow application aligns with business requirements and end-user expectations.
- **Focus Areas:**
    - Task management efficiency and usability.
    - User roles, permissions, and accessibility.
    - Validation of notifications, reminders, and reports.
- **Expected Outcome:** Approval from client representatives confirming readiness for production deployment.

5. Performance and Load Testing (Additional Testing Type)

- **Responsibility:** QA Performance Engineer
- **Tools Used:** Apache JMeter
- **Objective:** Assess system responsiveness, stability, and scalability under varying user loads.
  Focus Areas:
    - Simulate 50–500 concurrent users.
    - Measure average response times and server throughput.

- **Expected Outcome:** The application maintains acceptable performance metrics (≤2 seconds average response time).

6. Regression Testing

- **Responsibility:** QA Automation Engineer
- **Tools Used:** Selenium WebDriver, GitHub Actions (for CI automation)
- **Objective:** Ensure that new updates or bug fixes do not introduce new defects in existing modules.
- **Expected Outcome:** No critical functionality is broken after code merges or updates.

8. Test Schedule

Activity	Start Date	End Date	Responsible
Unit Testing	21 Oct 2025	23 Oct 2025	Developers
Integration Testing	24 Oct 2025	26 Oct 2025	QA Team

System Testing	27 Oct 2025	29 Oct 2025	QA Team
UAT	30 Oct 2025	31 Oct 2025	QA + Client
Test Closure	1 Nov 2025	2 Nov 2025	QA Lead

9. Resources

Resource Type	Description
Team Members	2 Developers, 2 Testers, 1 Project Manager
Tools	Google Docs (for collaboration), Jira (bug tracking), GitHub (versioning)
Hardware	Standard laptops with 8GB+ RAM
Software	Node.js, MongoDB, React, Selenium, Postman

10. Risks and Mitigation

Risk	Impact	Mitigation
Server downtime	High	Use backup EC2 instance
API failure	Medium	Enable logging and fallback mechanisms
Data inconsistency	Medium	Perform regular DB validation

Internet disruption	Low	Use Google Docs offline sync