Test Plan for Task Management Web Application

Author: Sam Lopez
Version: v1.0

Objective

The goal of this test plan is to outline the overall testing approach for validating the quality, functionality, and reliability of the Task Management Web Application. This plan ensures that all core features meet the business requirements and function correctly across user roles and scenarios.

Scope of Testing

In-Scope:

- Task creation, assignment, and prioritization
- Due date and milestone setting
- Calendar integration and task scheduling
- Survey creation and task-based workflows
- Dashboard summary for tasks (active, completed, overdue)
- Email and in-app notifications
- Search and filtering functionality
- Role-based access control (Admin, Manager, Contributor)
- Commenting and document upload on tasks
- Progress bar visualization for tasks

Out-of-Scope:

- Third-party email delivery systems (beyond our notification trigger)
- Mobile responsiveness (unless specified separately)
- Performance/load testing (covered in a different phase)

Testing Strategy

Functional Testing

- Validate core workflows like task creation, assignment, and editing
- Ensure role-based actions work correctly per user type
- Confirm task lifecycle changes reflect accurately across the system

UI/UX Testing

- Check alignment, consistency, and responsiveness of UI components
- Verify intuitive navigation between dashboard, tasks, and calendar views

Integration Testing

- Test integrations between the calendar, notifications, and survey modules
- Ensure backend services and frontend communication work as expected

Security Testing

- Validate role permissions, unauthorized access, and data privacy

Regression Testing

- Ensure new changes do not break existing functionality using automation suites

Test Environment

- Frontend: ReactJS Web Application (or applicable framework)
- Backend: Node.js/Express (assumed)
- Database: SQL / MongoDB
- Browsers: Chrome, Firefox, Edge (latest versions)
- Devices: Desktop (Windows/macOS), basic tablet check
- Environments: QA, Staging (pre-production)

Test Deliverables

- Test Scenarios & Test Cases (manual and automated)
- Test Data for all relevant modules
- Test Execution Reports
- Bug Reports with reproduction steps
- Final Test Summary Report

Roles and Responsibilities

Role	Responsibility
QA Lead	Test planning, strategy, reporting
QA Engineer	QA Engineer Test case design, execution, bug reporting
Automation Engineer	Build and maintain automated regression tests
Developer	Resolve issues, support in defect triage
Product Owner	Approve test scenarios and feature readiness
Scrum Master	Drive scrum ceremonies
Business Analyst	Create and Organize backlogs
Devops	Support CICD Integration and automation workflows

Risk Management

Risk	Mitigation Strategy
Late requirement changes	Frequent syncs with product team
Incomplete role permissions coverage	Early access to user-role matrix
Overlooked edge cases	Peer reviews of test cases and Test Case coverage with POs
Delayed builds for QA	Predefined build release window

Entry and Exit Criteria

Entry Criteria:

- Feature requirements are finalized and signed off
- QA environment is stable and accessible
- Test data is prepared
- Functional code is deployed to QA

Exit Criteria:

- All critical test cases are executed
- Major defects are resolved or deferred with sign-off
- Test coverage meets the agreed benchmark
- Regression suite passes without major blockers

Tools & Resources

Category	Tool
Test Management	Azure DevOps
Automation Framework (UI and API)	Playwright
Bug Tracking	Azure DevOps
CI/CD	Azure DevOps/Github Actions
Communication	Microsoft Teams
Programming language	Typescript/Javascript

Schedule Estimate

Phase	Duration
Requirement Analysis	4 Days
Test Case Design	6 Days
Test Execution	12 Days
Regression Testing	6 Days
Reporting and Sign-off	1 Day