# Quality Assurance (QA) Plan

|  |  |
| --- | --- |
| Task | Responsible Team Member |
| Quality Assurance Strategy | Ahmet Kaan Tırhış |
| Quality Factors & Metrics | Ahmet Kaan Tırhış |
| Test Plan | Ahmet Kaan Tırhış, Demir Doruk Dilek, Tuna Yılmaz |

## 1. Quality Assurance Strategy

### Overview

The quality assurance strategy ensures that PlanEasy meets functional and non-functional requirements while maintaining stability, usability, and performance. The testing process will involve both manual and automated testing methodologies to identify and eliminate potential issues before deployment.

### Testing Methodologies

- Unit Testing: Individual components (Task Management, Notifications) will be tested in isolation.

- Integration Testing: Ensuring smooth interaction between different system modules, such as UI and database operations.

- Functional Testing: Verifying that core features work as expected based on functional requirements.

- Usability Testing: Testing the UI for ease of use and intuitiveness.

- Performance Testing: Measuring response time, resource usage, and scalability.

- Security Testing: Ensuring user data protection and encryption methods.

### Automated vs. Manual Testing

|  |  |  |
| --- | --- | --- |
| Testing Type | Automated | Manual |
| Unit Testing | YES | NO |
| Integration Testing | YES | NO |
| Functional Testing | YES | YES |
| Usability Testing | NO | YES |
| Performance Testing | YES | NO |
| Security Testing | YES | NO |

## 2. Quality Factors & Metrics

Each quality factor will be measured using specific metrics to ensure system efficiency and reliability.

|  |  |  |
| --- | --- | --- |
| Quality Factor | Description | Measurement Metric |
| Performance | The system should respond quickly to user actions. | Average response time (ms) |
| Security | User data should be protected from unauthorized access. | Number of detected security vulnerabilities |
| Usability | The application should be easy to navigate and use. | User satisfaction survey score |
| Maintainability | The system should be easy to modify and expand. | Code complexity score (Cyclomatic Complexity) |

## 3. Test Plan

### Test Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Test Scenario** | **Description** | **Expected Outcome** | **Test Type** |
| **TC-001** | Task Creation | This test case verifies the ability of a user to create a new task with relevant details, including title, description, and due date. The system's response to saving the task and ensuring data persistence after application restart is evaluated. | The newly created task should be visible in the task list with accurate details. Upon restarting the application, the task should persist in the database without data loss. | Functional |
| **TC-002** | Task Modification | This test evaluates whether an existing task can be successfully updated with new values for title, description, or due date. The integrity of modified data after saving and reloading the application is also tested. | The task's modified attributes should be correctly saved and displayed without data loss. | Functional |
| **TC-003** | Task Deletion | This test case assesses the system’s ability to remove a task permanently from both the UI and the database. It verifies that deleted tasks do not reappear after application relaunch. | The task should be permanently removed from the UI and database, ensuring no residual data remains. | Functional |
| **TC-004** | System Performance Under Load | This test simulates high-load conditions by inserting 1000+ tasks and measuring the application’s response time, UI fluidity, and system resource usage. Performance degradation and memory leaks are monitored. | The application should remain responsive, with task operations executing in under **500ms**. No significant increase in memory consumption or crashes should occur. | Performance |
| **TC-005** | Unauthorized Access Prevention | This test evaluates whether an unauthenticated user can bypass authentication measures to access task data, modify records, or inject malicious SQL queries. The effectiveness of authentication and data protection mechanisms is assessed. | Unauthorized users should be blocked from accessing task-related data. Authentication and data validation layers should prevent unauthorized modifications or security breaches. | Security |

### Bug Tracking & Reporting

- GitHub Issues will be used to report and track bugs.

- Each bug report will include:

- Bug description

- Steps to reproduce

- Expected vs actual behavior

- Severity level

- Fixes will be assigned to developers, and updates will be documented through commits.