# Test Plan

---

## 1. Introduction

**Objective:**
This document outlines a comprehensive test plan for the Student Progress Tracking System. The system tracks and analyses student progress through goals, trials, and other performance metrics. This test plan covers functional, UI, performance, and security testing, including testing of new graphing functionalities that visually display goal results.

---

## 2. Scope

This test plan covers:

- **Functional Testing**: Validation of core features (goal management, trial management, notes, and graph generation).
- **Graph Functionality Testing**: Testing graphs that visually display results for various goals, ensuring correctness, readability, and relevance.
- **UI Testing**: Ensuring that the system's interface is intuitive, responsive, and consistent across platforms.
- **Performance Testing**: Verifying system performance under normal and stressed conditions.
- **Security Testing**: Ensuring proper user authentication, authorization, and data protection.
- **Cross-Browser/Device Testing**: Ensuring compatibility across multiple browsers and devices.
- **Regression Testing**: Verifying that new updates don't introduce defects to existing functionality.

---

## 3. Test Objectives

- Ensure goals, trials, notes, and graph generation work as expected.
- Validate that the graph functionality displays correct representations based on goal results.
- Confirm UI responsiveness and visual consistency across various devices and browsers.
- Identify any security vulnerabilities, particularly related to user data and system access.
- Test performance under different loads, including goal and graph data generation.
- Perform regression tests to ensure that new changes do not break existing features.

---

## 4. Test Strategy

- **Manual Testing**: Primarily used for UI, exploratory, and graph testing.
- **Automated Testing**: For regression testing and repetitive functional test cases.
- **Performance Testing**: Using tools like **JMeter** to simulate user load and test scalability.
- **Security Testing**: Tools such as **OWASP ZAP** for security assessments like SQL injection, XSS, and CSRF.
- **Graph Testing**: Manually verify correct graph generation, ensuring different graph types (e.g., line, bar, pie) represent the goal results accurately.

---

## 5. Test Environment

- **Browsers**: Google Chrome, Firefox, Microsoft Edge, Safari (latest versions).
- **Operating Systems**: Windows 10/11, macOS, iOS, Android.
- **Devices**: Desktop, Laptop, Tablets (iPad, Android), Smartphones (iPhone, Android).
- **Automation Tools**: Selenium WebDriver (for UI and functional automation), JMeter (for performance testing).
- **Security Tools**: OWASP ZAP (for security testing).

---

## 6. Test Deliverables

1. **Test Plan**: A document outlining the test strategy, objectives, scope, and schedule.
2. **Test Cases**: Detailed cases covering functional, UI, graphing, and security tests.
3. **Test Scripts**: Automation scripts for repetitive tests, especially around regression.
4. **Test Results**: Documented results of all test cases and automation scripts.
5. **Defect Reports**: Bug logs categorized by severity, including screenshots and reproduction steps.
6. **Test Summary Report**: A final summary of test outcomes, issues identified, and areas of risk.

---

## 7. Test Execution Schedule

| Phase | Start Date | End Date | Duration | Resources |
|---|---|---|---|---|
| Test Planning | Day 1 | Day 3 | 3 Days | QA Lead |
| Test Case Development | Day 4 | Day 8 | 5 Days | QA Engineers |
| Environment Setup | Day 6 | Day 10 | 5 Days | DevOps |
| Smoke Testing | Day 11 | Day 13 | 3 Days | QA Engineers |
| Functional Testing | Day 14 | Day 21 | 8 Days | QA Engineers |
| Graph Functionality Testing | Day 14 | Day 21 | 8 Days | QA Engineers |
| Performance & Load Testing | Day 22 | Day 26 | 5 Days | QA Engineers |
| Security Testing | Day 27 | Day 30 | 4 Days | Security Team |

| | | | | |
|---|---|---|---|---|
| Regression Testing | Day 31 | Day 33 | 3 Days | QA Engineers |
| Final Report | Day 34 | Day 35 | 2 Days | QA Lead |

---

**8. Test Cases**

## 8.1 Functional Testing

### 8.1.1 Goal Management

- **Test Case 1.1: Add New Goal**
  - **Pre-condition**: User is logged in and on the goal dashboard.
  - **Steps**:
    1. Click "Add Goal".
    2. Enter goal details (e.g., "Complete 5 Math Assignments").
    3. Save the goal.
  - **Expected Result**: Goal is added successfully and is displayed on the dashboard.
- **Test Case 1.2: Edit Goal**
  - **Pre-condition**: At least one goal exists.
  - **Steps**:
    1. Select a goal.
    2. Edit the goal details (e.g., modify the target).
    3. Save the changes.
  - **Expected Result**: Goal details are updated successfully.
- **Test Case 1.3: Delete Goal**
  - **Pre-condition**: At least one goal exists.
  - **Steps**:
    1. Select a goal.
    2. Click "Delete".
  - **Expected Result**: The goal is removed from the dashboard.

### 8.1.2 Trial Management

- **Test Case 2.1: Record Trial Results**
  - **Pre-condition**: A goal with trials exists.
  - **Steps**:
    1. Open the trial screen for a specific goal.
    2. Enter trial results (e.g., correct, incorrect, cue).
    3. Save the trial results.
  - **Expected Result**: Results are saved and reflected in the trial history.
- **Test Case 2.2: Reset Trial**
  - **Pre-condition**: A completed trial exists.
  - **Steps**:
    1. Select the trial.
    2. Click "Reset".
  - **Expected Result**: Trial results are reset to an empty state.

### 8.1.3 Notes Management

- **Test Case 3.1: Add New Note**
  - **Pre-condition**: The user is viewing a specific goal.
  - **Steps**:
    1. Click "Add Note".
    2. Enter note details.
    3. Save the note.
  - **Expected Result**: The note is saved and displayed under the goal.
- **Test Case 3.2: Edit Existing Note**
  - **Pre-condition**: A note exists.
  - **Steps**:
    1. Select the note.
    2. Modify the content.
    3. Save the note.
  - **Expected Result**: The note is updated with the new content.

---

## 8.2 Graph Functionality Testing

### 8.2.1 Generate Graphs Based on Goal Results

- **Test Case 4.1: Display Line Graph for Goal Results**
  - **Pre-condition**: A goal has sufficient trial data.
  - **Steps**:
    1. Select a goal.
    2. Click "Graph" and select "Line Graph".
  - **Expected Result**: A line graph is displayed showing progress over time.
- **Test Case 4.2: Display Bar Graph for Trial Results**
  - **Pre-condition**: A goal has trial results.
  - **Steps**:
    1. Select a goal.
    2. Click "Graph" and choose "Bar Graph".
  - **Expected Result**: A bar graph is displayed with correct data reflecting different trial results.
- **Test Case 4.3: Display Pie Chart for Goal Performance**
  - **Pre-condition**: A goal has various outcomes (correct, incorrect, cue).
  - **Steps**:
    1. Select a goal.
    2. Click "Graph" and select "Pie Chart".
  - **Expected Result**: A pie chart displays the percentage of correct, incorrect, and cued responses.
- **Test Case 4.4: Graph Interactivity (Hover Information)**
  - **Pre-condition**: Any graph is generated.
  - **Steps**:
    1. Hover over data points in the graph.
  - **Expected Result**: Tooltips with detailed information (e.g., date, trial number, result) appear when hovering over points.

## 8.3 User Interface (UI) Testing

### 8.3.1 Responsive Design

- **Test Case 5.1: Check Layout on Desktop**
  - **Steps**: Open the application on a desktop browser.
  - **Expected Result**: All UI elements are visible, properly aligned, and functional.
- **Test Case 5.2: Check Layout on Mobile**
  - **Steps**: Open the application on a mobile device.
  - **Expected Result**: UI adapts to smaller screens and remains fully functional.

### 8.3.2 Accessibility Testing

- **Test Case 5.3: Keyboard Navigation**
  - **Steps**: Navigate through the application using only the keyboard.
  - **Expected Result**: All buttons and fields are accessible without using a mouse.
- **Test Case 5.4: Screen Reader Compatibility**
  - **Steps**: Use a screen reader to navigate and interact with the application.
  - **Expected Result**: Screen reader correctly reads out content, labels, and tooltips.

---

### 8.4 Performance Testing

- **Test Case 6.1: Goal Load Performance**
  - **Steps**: Load a large number of goals and track how quickly they are displayed.
  - **Expected Result**: The system handles and displays goals within acceptable limits (e.g., < 3 seconds).
- **Test Case 6.2: Graph Generation Performance**
  - **Steps**: Generate a graph for a goal with a large amount of data.
  - **Expected Result**: Graph generation occurs within an acceptable time (e.g., < 2 seconds).

---

### 8.5 Security Testing

- **Test Case 7.1: Unauthorized Access**
  - **Steps**: Attempt to access a restricted page without logging in.
  - **Expected Result**: The system redirects the user to the login page.
- **Test Case 7.2: SQL Injection**
  - **Steps**: Input SQL commands into form fields.
  - **Expected Result**: The system handles the input safely and does not execute any SQL commands.

---

## 9. Risks & Mitigations

- **Risk**: Performance issues with large data sets (especially for graphs).
  - **Mitigation**: Perform load testing and optimize queries for graph generation.
- **Risk**: UI inconsistencies across different devices.
  - **Mitigation**: Thorough cross-browser and device testing with manual and automated tools.

---

## 10. Conclusion

This test plan ensures comprehensive coverage of the Student Progress Tracking System. Emphasis is placed on functional correctness, performance, graph accuracy, and security. The system will undergo rigorous testing to ensure all features, especially the graphing feature, meet user expectations for performance and accuracy.