# Software Test Description (STD) Template for Discord App

---

---

**Project Name**: Discord
**Version**: 1.0
**Test Documentation Version**: 1.0
**Date**: [18/02/2025]
**Prepared by**: [Mulugeta Beiene]
**Reviewed by**:
**Approval**:

---

## 1. Introduction

### 1.1 Purpose

The purpose of this Software Test Description (STD) is to provide detailed test information for validating the functionality, performance, security, and usability of the Discord app. The goal is to ensure that Discord functions as per the specified requirements, is free of defects, and provides a seamless experience across multiple platforms.

### 1.2 Scope

*This STD will cover*:

- **Functional Testing**: Verifying that all core features of Discord are working as expected.
- **Non-Functional Testing**: Performance, security, and compatibility testing to ensure stability under various conditions.
- **Regression Testing**: Ensuring that bug fixes and new feature additions do not break existing functionality.

- **Usability Testing**: Ensuring that the app is user-friendly and intuitive.
- **Security Testing**: Verifying that the app adheres to security best practices.

### 1.3 Test Case Identification

Each test case will be identified with a unique ID to ensure traceability. The format for the test case ID will be **"DISCORD-"**, where is a unique number.

---

## 2. Test Approach

### 2.1 Test Methodology

The testing approach will follow a structured methodology, including the following:

- **Manual Testing**: To check user interactions and UI-based functionality.
- **Automated Testing**: For repetitive tasks such as UI regression tests using tools like **Selenium**.
- **Load and Performance Testing:** Using tools like **JMeter** to simulate high user traffic and identify performance bottlenecks.
- **Security Testing:** Using tools like **OWASP ZAP** to identify security vulnerabilities such as SQL injections, XSS, and CSRF attacks.

---

## 3. Test Types and Focus Areas

## 3.1 Functional Testing

Functional tests will focus on validating the core features of Discord:

- **User Authentication**:
  - Test login/logout functionality.
  - Test account creation and password recovery.
  - Validate multi-factor authentication.
- **Voice and Text Communication**:
  - Test sending and receiving text messages in channels and direct messages.
  - Test voice call functionality, including joining/leaving channels, muting, and push-to-talk.
  - Test group voice calls, ensuring no lag and clear audio.
- **Server Management**:
  - Test server creation, deletion, and modification.
  - Test user permissions within servers (e.g., admin, moderator, member roles).
  - Validate bot integration and functionality (e.g., moderation bots).
- **File Sharing**:
  - Test uploading and downloading files in chat.
  - Validate media types (images, videos, GIFs) supported for sharing.
- **Notifications**:
  - Test notification preferences (muting channels, server notifications).

    - Test push notifications on mobile devices for direct messages and server activities.

---

## 3.2 Non-Functional Testing

- **Performance Testing**:
    - Load Test: Simulate multiple users (e.g., 5000+ users) joining a voice channel and testing for any degradation in performance.
    - Stress Test: Test the app under extreme conditions to see if it crashes or becomes unresponsive.
    - Latency Test: Measure the latency of voice messages and the time taken for text messages to be sent/received.
- **Security Testing**:
    - Test for vulnerabilities such as SQL injection, XSS, and session hijacking.
    - Verify the encryption of messages and voice data, ensuring user data privacy.
    - Test for authentication weaknesses (e.g., brute force login attempts, session management).
- **Compatibility Testing**:
    - Test Discord's functionality across multiple operating systems:windows,lenix, iOS, and Android.
    - Test for cross-browser compatibility (Chrome, Firefox, Safari, Edge).
    - Test various screen sizes and resolutions on mobile devices (smartphones, tablets).

---

## 3.3 Usability Testing

- **Ease of Navigation**:
  - Validate that users can easily navigate through the app (desktop and mobile versions).
  - Ensure that features are easily accessible, with no confusing interfaces or settings.
- **UI/UX Consistency**:
  - Verify the consistency of UI elements (buttons, text fields, icons) across platforms.
  - Check for mobile responsiveness, ensuring elements adjust properly for smaller screens.
- **Error Handling**:
  - Test for clear error messages when things go wrong (e.g., network failures, permissions issues).
  - Ensure users are provided with helpful instructions if something fails (e.g., failed login, server not found).

---

## 4. Test Case Design

### 4.1 Test Case Format

Each test case will include the following details:

- **Test Case ID**: Unique identifier for the test case.
- **Test Case Name**: Brief description of the test case.
- **Test Objective**: What is being validated in the test.

- **Pre-Conditions**: Any setup or prerequisites for the test.
- **Test Steps**: Detailed steps to execute the test case.
- **Expected Results**: What the expected outcome is.
- **Pass/Fail Criteria**: Conditions for passing or failing the test case.

---

## 4.2 Example Test Case:

**Test Case ID**: DISCORD-001
**Test Case Name**: User Login Test
**Test Objective**: To validate that users can successfully log into the Discord app with correct credentials.
**Pre-Conditions**: User has a valid account and the app is installed.
**Test Steps**:

1. Open the Discord app.
2. Enter valid username and password.
3. Click on the "Login" button.
4. Wait for the app to load the main dashboard. **Expected Results**:

- User is successfully logged in and directed to the home screen.
- User’s profile and server information appear correctly.
  **Pass/Fail Criteria**:
- **Pass**: User successfully logs in and can interact with the app.
- **Fail**: User cannot log in, or there is an error on the login screen.

---

## 5. Test Execution & Logging

## 5.1 Test Execution

The tests will be executed based on the defined schedule, with each test case being executed by the assigned tester. Results will be logged for each test case, and defects will be logged for any failed test case.

## 5.2 Defect Logging

Any defect found during testing will be logged in the defect tracking system (e.g., Jira), with the following details:

- **Defect ID**: Unique identifier for the defect.
- **Summary**: Brief description of the defect.
- **Severity**: Severity level of the defect (Critical, High, Medium, Low).
- **Steps to Reproduce**: How the defect was encountered.
- **Status**: Current status (Open, In Progress, Closed).

---

## 6. Risks and Mitigations

### 6.1 Risks

- ***Inconsistent Performance on Low-End Devices***: Discord may behave differently on lower-end devices, especially during voice chats or high server loads.
- ***Security Vulnerabilities*:** Possible flaws in user data protection mechanisms could lead to potential breaches.

## 6.2 Mitigation Plan

- ***Performance Optimization***: Focus on optimizing resource usage on low-end devices.
- ***Continuous Security Audits***: Regularly run security scans to identify vulnerabilities and patch them promptly.

---

## 7. Approval and Sign-Off

### Test Case Approval:

| Name | Role | Date | Signature |
|---|---|---|---|
| [Test Lead Name] | Test Lead | [Date] | [Signature] |
| [QA Manager Name] | QA Manager | [Date] | [Signature] |
| [Product Owner Name] | Product Owner | [Date] | [Signature] |