1. Test Planning

Objective:
To verify that the Discord Desktop App meets user expectations for performance, functionality, and stability across its main communication features. The app should facilitate smooth messaging, audio, and video services, ensure secure data handling, and provide a cohesive user experience.

Scope

Features to Be Evaluated:

1. User Registration & Login:
    - Validate the registration and login process for both new and returning users.
    - Ensure secure authentication, including 2-Factor Authentication (2FA).
    - Test responses to incorrect or expired login credentials.
2. Messaging (Direct and Group Chats):
    - Test functionality for sending and receiving text and multimedia messages.
    - Measure performance for:
        - Text-only messages.
        - Media such as images, videos, and documents.
    - Verify seamless file, link, and data sharing between users.
3. Notifications (Real-Time Alerts):
    - Assess the notification system's ability to alert users about messages, mentions, and reactions.
    - Ensure notifications are timely and customizable within settings.
4. Audio and Video Quality:
    - Examine the clarity and stability of audio/video calls.
    - Ensure the app maintains quality during transitions from text chat to video calls.
    - Test call stability across varying network conditions.

5. User Interface (UI) Accessibility:
    - Verify consistency in design across different operating systems (Windows, macOS, Linux).
    - Ensure UI elements like buttons, text fields, and menus are accessible and identifiable.
6. Profile and Settings Management:
    - Test profile functionalities, including profile picture update, bio editing, and status customization.
    - Verify settings for notifications, privacy, and user preferences.
7. AI and Bot Integration:
    - Validate the integration of AI bots for enhanced communication features.
    - Ensure users can easily enable or disable bots.
8. User Status Indicators:
    - Confirm functionality of online status indicators (e.g., online, offline, idle).
    - Test typing indicators and last-seen visibility and accuracy.
9. Security and Data Protection:
    - Verify message encryption for user privacy.
    - Confirm other security measures, such as end-to-end encryption, XSS, and SQL injection protection.
10. Multi-Server Management and Subthreads:
    - Ensure users can create and organize channels within servers.
    - Test subthread functionality to support topic-specific discussions within larger groups.

Features Not to Be Tested:

- Third-party Integrations: Exclude features like Apple Music integration.
- Mobile-Specific Enhancements: Focus remains on the desktop version’s usability and design.

2. Test Strategy

Manual Testing:

- Objective: Ensure core functionalities like messaging, audio, and video calls work seamlessly.
- Focus Areas: Real-time messaging, consistent UI, and profile settings will be manually verified.

Automated Testing:

- Smoke Testing: Automate essential tasks (login, message sending/receiving, notifications).
- Regression Testing: Verify existing functions after updates to maintain stability.

Testing Tools:

- Selenium/Appium: Cross-platform automated testing.
- JMeter: Performance and load testing.
- Figma: UI design validation.

3. Test Design Techniques

1. Equivalence Partitioning
    - Inputs include usernames, passwords, message content, and server/channel configurations.
    - Test valid, invalid, and boundary scenarios (e.g., empty fields, incorrect passwords).
2. Boundary Value Analysis
    - Test limits for fields like username length, message size, and group chat participant count.
3. Decision Table Testing
    - Examine combinations of conditions such as notification settings, online/offline status, and message types.
4. State Transition Testing
    - Track user states, such as logged in, typing, sending, and receiving messages.

4. Real Test Cases

Test Case 1: Verify Direct Message Delivery

- Test Case ID: TC_001
- Description: Ensure a text message is sent and delivered successfully in a direct message.
- Preconditions:
    - User is logged in to Discord Desktop App.
    - Internet connection is active.
- Test Steps:

1. Open the Discord Desktop App.
2. Navigate to the “Direct Messages” section.
3. Select a contact from the list.
4. Type "Hello" in the text box and press Enter.

- Expected Results: The message should display in the chat window with a "Delivered" status.
- Actual Results: To be completed after testing.
- Status: Pass/Fail

Test Case 2: Verify Profile Picture Update

- Test Case ID: TC_002
- Description: Ensure a user can update their profile picture successfully.
- Preconditions:
    - User is logged in to Discord.
    - New profile picture file is accessible on the desktop.
- Test Steps:

1. Open profile settings.
2. Click on the profile picture.
3. Select a new image file from the local drive.
4. Confirm and save changes.

- Expected Results: Profile picture updates to the new image.
- Actual Results: To be completed after testing.
- Status: Pass/Fail

Test Case 3: Verify Group Chat Creation

- Test Case ID: TC_003
- Description: Confirm that a user can create a group chat with multiple members.
- Preconditions:
    - User is logged in to Discord Desktop App.
    - At least three contacts are available.
- Test Steps:

1. Click “Create Group DM.”
2. Select three contacts to add to the group.
3. Name the group and click “Create.”

- Expected Results: Group chat is created, and all members can communicate.
- Actual Results: To be completed after testing.
- Status: Pass/Fail

Test Case 4: Verify Notification Sound for New Messages

- Test Case ID: TC_004
- Description: Confirm that the notification sound plays when a new message is received while the app is minimized or inactive.
- Preconditions:
    - User is logged in to the Discord Desktop App.
    - Notifications are enabled, and the app is minimized.
- Test Steps:

1. Minimize the Discord Desktop App.
2. Send a test message to this account from another device or account.

- Expected Results: The user should hear a notification sound alerting them of the new message.
- Actual Results: To be filled after testing.
- Status: Pass/Fail

Test Case 5: Verify File Sharing Limit in Chat

- Test Case ID: TC_005
- Description: Ensure that an error message appears if a user tries to send a file exceeding the allowed size limit (e.g., 8MB).
- Preconditions:
    - User is logged in to the Discord Desktop App.
    - A file larger than 8MB is available on the desktop.
- Test Steps:

1. Open a direct message or group chat.
2. Attempt to attach and send a file larger than 8MB.

- Expected Results: An error message should appear indicating that the file size exceeds the allowed limit, preventing the upload.
- Actual Results: To be completed after testing.
- Status: Pass/Fail

5. Bug Reports

Bug Report 1: Notification Sound Fails in Inactive Mode

- Bug ID: BUG_001
- Summary: Notification sound does not play when a new message is received while the app is minimized.
- Steps to Reproduce:
    1. Log in to Discord Desktop App.
    2. Enable notification sounds.
    3. Minimize the app.
    4. Send a message from another device.
- Expected Result: Notification sound should play.
- Actual Result: No sound is heard.
- Severity: Medium
- Priority: High
- Attachments: Screenshot of settings.

Bug Report 2: Profile Picture Update Not Reflected

- Bug ID: BUG_002
- Summary: Profile picture fails to update after uploading a new image.
- Steps to Reproduce:
    1. Open Discord profile settings.
    2. Upload and save a new profile picture.
    3. Check profile picture in chat.
- Expected Result: Profile picture updates.
- Actual Result: Picture remains unchanged.
- Severity: Low
- Priority: Medium
- Attachments: Screenshot showing unchanged profile picture.

Bug Report 3: Inconsistent Message Delivery Status

- Bug ID: BUG_003
- Summary: The message delivery status remains "Sent" instead of updating to "Delivered" or "Read" after the recipient has viewed it.
- Steps to Reproduce:
    1. Log in to the Discord Desktop App.
    2. Send a message to a contact who is online.
    3. Instruct the recipient to open and read the message.
- Expected Result: The delivery status should update to "Delivered" or "Read" upon being read by the recipient.
- Actual Result: The status stays as "Sent" even after the recipient reads the message.
- Severity: Medium
- Priority: High
- Attachments: Screenshot showing the incorrect delivery status.

Bug Report 4: File Upload Exceeding Limit Not Restricted

- Bug ID: BUG_004
- Summary: Discord permits users to upload files that exceed the maximum allowed size without displaying a warning or error.
- Steps to Reproduce:
    1. Open a direct message in the Discord Desktop App.
    2. Attach a file larger than 8MB.
    3. Attempt to send the file.
- Expected Result: An error or warning message should appear, preventing the upload of files that exceed the size limit.
- Actual Result: The file uploads successfully without any warnings, bypassing the size restriction.
- Severity: High
- Priority: High
- Attachments: Screenshot showing a successful upload of an oversized file.

6. Test Execution

Functional Testing:

- Evaluate core functions like text/media messaging, voice/video calls, notifications, and profile management.
- Ensure stability during feature transitions.

UI Testing:

- Verify that all UI components are compliant with the design and accessible.
- Confirm consistency across platforms (Windows, macOS, Linux).

Non-Functional Testing:

- Performance Testing: Simulate multiple users in high-load situations.
- Usability Testing: Assess the interface's ease of use and intuitiveness.
- Security Testing: Check for vulnerabilities like SQL injection and XSS.

7. Test Environment

Hardware Requirements:

- Desktops: Systems with Windows, macOS, and Linux.
- Network: Test on various internet speeds.

Software Requirements:

- Browsers: Compatibility testing across Chrome, Firefox, and Edge.
- Testing Tools: JIRA for issue tracking, Selenium for automation, Postman for API testing, and JMeter for load testing.

8. Test Deliverables

1. Test Cases and Scripts: Comprehensive scenarios covering all essential features.
2. Bug Reports: Detailed descriptions of identified issues.
3. Test Results Summary: Documented results with issues highlighted.
4. Performance Metrics: Data on app performance under load.
5. Security Reports: Analysis of vulnerabilities.

9. Risks and Mitigation

- Risk: High latency or app crashes under heavy load.
    - Mitigation: Perform thorough load testing under different network conditions.
- Risk: Interface complexity for new users.
    - Mitigation: Include user-friendly tooltips and a guided tour for beginners.