Crypto Trading Bot - Test Plan

1. Introduction

This document outlines the testing strategy for the "Crypto Trading Bot" application. The goal is to verify that all functional requirements are met, the system is stable, and the integration between the frontend (React/Next.js), backend (Flask API), and the MT5 trading platform is seamless.

2. Test Objectives

- Verify all functional requirements as specified in the project plan.
- Ensure data integrity for user configurations, trade history, and market data.
- Validate the successful integration and communication between the frontend, backend API, and MT5.
- Test the trading execution logic, including stop-loss and take-profit calculations.
- Confirm the application's UI is responsive and compatible across target devices.
- Identify and track defects to resolution.

3. Scope

In-Scope Features:

- **User Authentication:**
  Firebase authentication and user session handling.
- **Market Data:**
  Live market data display and watchlist functionality.
- **News Feed:**
  Interface for displaying crypto news updates.
- **Bot Configuration:**
  UI screens for setting, saving, and updating trading bot parameters.
- **Backend API (Flask):**
  All endpoints for news, market data, user config, and trading.
- **MT5 Integration:**
  Connectivity and trading execution logic (buy/sell, stop-loss, take-profit).
- **Database:**
  Storage and retrieval of user configurations and trade history.
- **UI/UX:**
  Responsiveness, layout, and cross-device compatibility.

Out-of-Scope Features:

- Testing of the core Firebase authentication service (assumed to be stable).
- Testing of the MT5 platform itself (assumed to be stable).
- Testing the accuracy of the third-party news or market data feeds (we will only test that the data is *displayed* correctly).
- Comprehensive performance, load, or security penetration testing (beyond basic checks).

4. Test Strategy & Levels

Based on the project plan, testing will be conducted in the following phases:

- **Unit Testing (by Dev & QA):**
  - Testing individual Flask API endpoints.
  - Testing individual React/Next.js components.
  - Testing modular trading logic functions (e.g., stop-loss calculation).

- **Integration Testing (by QA):**

  - **Frontend <-> Backend:**
    Verify frontend UI actions correctly call the Flask API and handle responses (e.g., saving a bot configuration).
  - **Backend <-> MT5:**
    Verify the backend API can successfully connect to MT5 and execute trading commands.
  - **Backend <-> Database:**
    Verify user data and trade history are correctly saved and retrieved.

- **System Testing (by QA):**
  - End-to-end testing of the complete, integrated application.
  - Simulating user scenarios from login to bot configuration to trade execution.
  - Validating the synchronization of data across the entire system (e.g., a trade executed in MT5 is reflected in the UI's trade history).

- **Usability & Compatibility Testing (by QA):**
  Testing the UI on different browsers (Chrome, Firefox) and devices (desktop, mobile) to ensure responsiveness.

5. Key Personnel

- **QA:** Ali Asad
- **Development:** Waqar (Dev), Muhammad (Dev)
- **Stakeholders:** Waqar (PM), Management