TradeJournal SRS

Software Requirements Specification: TradeJournal MVP

- **Product Name:** TradeJournal
- **Version:** 1.0
- **Status:** Draft
- **Date:** August 9, 2025

---

1. Introduction

1.1 Purpose

This document provides a detailed specification of the requirements for the Minimum Viable Product (MVP) of the TradeJournal web application. It is intended for developers, QA testers, and project managers to understand the system's functionality, constraints, and behavior.

1.2 Scope

The software, **TradeJournal**, is a web-based platform that enables users to manually log their financial trades, add journal entries, and view key performance metrics via a dashboard. This document covers the core features for the MVP release: user authentication, trade logging, and basic analytics.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS:** Software Requirements Specification
- **PRD:** Product Requirements Document
- **MVP:** Minimum Viable Product
- **P&L:** Profit and Loss
- **UI:** User Interface
- **API:** Application Programming Interface
- **JWT:** JSON Web Token

1.4 References

- TradeJournal MVP - Product Requirements Document, v1.0

1.5 Document Overview

This document begins with a general description of the product and its constraints. Section 3 contains the heart of the document, detailing the specific functional, interface, non-functional, and database requirements for the development team.

---

2. Overall Description

2.1 Product Perspective

TradeJournal is a new, self-contained web application. It will be built from the ground up and does not depend on or integrate with any existing software. The system is composed of three main parts: a React frontend, a FastAPI backend API, and a PostgreSQL database.

2.2 Product Functions

The MVP will provide the following key functions:

1. **User Registration and Authentication:** Securely create and manage user accounts.
2. **Trade Logging:** Allow users to manually enter the details of their trades.
3. **Journaling:** Allow users to add notes and reflections to each trade.
4. **Performance Analytics:** Calculate and display core performance metrics on a dashboard.

2.3 User Characteristics

The target users are retail traders who are comfortable using modern web applications and are familiar with standard trading terminology. No specialized training is required to use the application.

2.4 Constraints

1. The system **shall** be developed using the specified technology stack: React (Frontend), Python/FastAPI (Backend), and PostgreSQL (Database).
2. The application **shall** be accessible via modern web browsers (latest versions of Chrome, Firefox, Safari, and Edge).
3. All user data **shall** be stored in a centralized PostgreSQL database.
4. The system **shall** operate stateless where possible, managing sessions via JWTs.

2.5 Assumptions and Dependencies

1. Users are assumed to have a stable, active internet connection.
2. The infrastructure (hosting provider, database service) is assumed to be reliable and maintained.

---

3. Specific Requirements

3.1 External Interface Requirements

- **User Interface:** The application shall provide a responsive web-based UI. The design will be clean, intuitive, and consistent across all pages. All functionality shall be accessible through this graphical interface.

3.2 Functional Requirements

Module: Authentication

- **FR-AUTH-001:** The system shall provide a user registration page with fields for an email address and a password.
- **FR-AUTH-002:** The system shall validate that the provided email is unique within the database.
- **FR-AUTH-003:** Passwords must be a minimum of 8 characters.
- **FR-AUTH-004:** The system shall provide a login page for existing users.
- **FR-AUTH-005:** Upon successful login, the API shall generate and return a JWT to the client for authenticating subsequent requests.
- **FR-AUTH-006:** The system shall provide a logout mechanism that invalidates the user's session on the client side.

Module: Trade Management

- **FR-TRADE-001:** An authenticated user shall be able to access a form to log a new trade.
- **FR-TRADE-002:** The trade entry form shall contain the following fields:
  - Ticker Symbol (Text, required)
  - Direction (Dropdown: 'Long' or 'Short', required)
  - Entry Date & Time (Date/Time Picker, required)
  - Exit Date & Time (Date/Time Picker, required)
  - Time Zone(Dropdown: all available timezones, required)
  - Entry Price (Numeric, required, positive value
  - Exit Price (Numeric, required, positive value)
  - Quantity (Integer, required, positive value)
- **FR-TRADE-003:** Upon submission, the backend shall calculate the trade's P&L and store it with the trade record.
- **FR-TRADE-004:** The system shall display all of a user's trades in a chronological, sortable list (the "Trade Log").
- **FR-TRADE-005:** Users shall be able to click a trade to view its full details and add/edit a text-based journal entry for it.

Module: Dashboard

- **FR-DASH-001:** The dashboard shall display analytics calculated *only* from the logged-in user's trades.
- **FR-DASH-002:** The system shall calculate and display the following metrics:
  - **Total P&L:** Sum of P&L from all trades.
  - **Win Rate:** (Number of trades with P&L > 0) / (Total Number of Trades) * 100%.
  - **Average Win:** Average P&L of all winning trades. If there are no winning trades, 'Average Win' shall display $0.00.
  - **Average Loss:** Average P&L of all losing trades. If there are no losing trades, 'Average Loss' shall display $0.00.

3.3 Non-Functional Requirements

- **NFR-PERF-001 (Performance):** API responses for fetching data (e.g., dashboard stats, trade log) shall complete in under 500ms for a user with up to 1,000 trades.
- **NFR-SEC-001 (Security):** All user-provided data must be sanitized on the backend to prevent injection attacks (e.g., SQLi, XSS).
- **NFR-SEC-002 (Data Security):** User passwords shall be salted and hashed using a strong, modern algorithm (e.g., bcrypt).
- **NFR-SEC-003 (Access Control):** A user shall only be able to view or modify data that they own. API endpoints must enforce this ownership check on every request.
- **NFR-USAB-001 (Usability):** The application interface must be responsive and fully functional on screen widths from 360px to 1920px.

3.4 Database Requirements

- The system shall use a PostgreSQL database. The schema will include at least the following tables with enforced relationships:
  - users
    - id (Primary Key)
    - email (Unique)
    - password_hash
    - created_at
  - trades
    - id (Primary Key)
    - user_id (Foreign Key to users.id)
    - ticker
    - direction
    - entry_price
    - exit_price
    - quantity
    - entry_timestamp
    - exit_timestamp
    - pnl
    - journal_notes (Text)
- Data integrity shall be enforced at the database level (e.g., a trade cannot be created without a valid user_id).
- All timestamps (`entry_timestamp`, `exit_timestamp`) will be converted to and stored in UTC in the database.