Functional Requirements Specification: TradeAdvance Stock Trading Platform

# 1. Introduction and Scope

TradeAdvance is a full-stack stock trading platform that provides real-time market insights, portfolio management, automated strategies, and AI-driven analysis. It is designed for retail traders and investors, offering features such as comprehensive stock search, live charting with technical indicators, automated strategy execution, portfolio dashboards, predictive forecasting (ARIMA), and stock recommendations (K-Means clustering). This Functional Requirements Specification (FRS) defines the functional and non-functional requirements for TradeAdvance. It specifies the features, constraints, user roles, data model, interfaces, and validation criteria that the system must satisfy. It is intended for stakeholders, business analysts, developers, and QA teams, and will serve as a basis for design, implementation, and testing.

# 2. System Overview

TradeAdvance is a web-based platform built on a modern architecture. The backend uses Python’s FastAPI framework, providing RESTful endpoints. The frontend is implemented with HTML, CSS (Semantic UI), and JavaScript, delivering interactive dashboards and forms. An SQLite3 database stores user accounts, portfolio data, historical prices, and strategy configurations. TradeAdvance integrates with the Alpaca Markets API for real-time stock data and order execution. Internally, it uses machine-learning models (ARIMA for forecasting and K-Means for clustering) to generate predictions and recommendations. The system is hosted on standard web infrastructure (e.g. Linux server or cloud service) and is accessible via modern web browsers on desktop and mobile devices.

## Key technologies and environment include:

• Backend: Python, FastAPI, RESTful API endpoints

• Frontend: HTML5, CSS, JavaScript (Semantic UI framework)

• Database: SQLite3 (local or cloud-hosted) for data storage

• External APIs: Alpaca Markets API for market data and trading; optional email/SMS service for notifications

• Data Storage: Local file system or cloud for storing static charts and logs

• Deployment: Platform can be deployed on standard cloud (AWS, Azure) or on-premise web server, using secure HTTPS connections.

# 3. Actors and User Roles

• Trader (End User): A registered individual who uses the platform to manage and trade investments. The Trader can create an account, log in, view portfolio insights, search stocks, execute trades (manually or via strategies), view forecasts/recommendations, and review transaction history.

• Administrator: (Optional role) A system administrator who can manage user accounts, set system-wide parameters, and maintain the platform (backup/restore). Not visible in the UI for end users, but implied for maintenance.

• External Services: Alpaca Markets API and email/notification services act as system interfaces (no direct user role).

Each Trader must authenticate (login) before accessing private features. Authentication and authorization requirements are fundamental to the system. Different users may have different access levels (e.g. Admin vs. Trader), controlling who can perform certain actions.

# 4. Assumptions and Constraints

• Market Coverage: Initially, the platform supports U.S. equity markets (e.g. NASDAQ, NYSE). It is assumed that Alpaca’s data covers these markets. Expansion to international markets is future scope.

• Data Availability: Real-time price data is assumed available via Alpaca API. The system assumes Alpaca’s uptime and rate limits; if data is unavailable, the system must display an error message to the user.

• User Requirements: Users are assumed to have internet access and a modern web browser. The system assumes users will not attempt to exceed API rate limits or perform prohibited trading (subject to Alpaca and regulatory rules).

## System Constraints:

• The database is limited by SQLite; expected to handle up to tens of thousands of stock records and concurrent users. For very large scale, migrating to a more robust DB may be required in future.

• Trade execution is limited by the API’s rules (e.g. trading hours, account balance, buying power).

• All sensitive data (user credentials, API keys) are stored securely (passwords hashed; API keys in server environment).

## Dependencies:

The platform depends on continuous availability of Alpaca API for market data/trading and email services for notifications.

## Error Handling:

The system will display user-friendly error messages when operations fail (e.g. invalid login, insufficient funds). All errors and exceptions will be logged for debugging. Requirements for error messaging and logging are defined later.

# 5. Functional Requirements

| Requirement ID | Description | Validation |
| --- | --- | --- |
| FR-1: User Registration | The system shall allow a new user to register by providing a unique username, email address, and password. It shall validate that required fields are present and that the email is not already registered. | Upon successful registration, a new user account is created and the user is redirected to login. If the email/username exists, an error 'Email or Username already exists' is displayed. |
| FR-2: User Login | The system shall allow a registered user to log in by entering valid credentials (username/email and password). | On successful authentication, the user is granted access to the platform (portfolio dashboard). Invalid credentials must be rejected with 'Invalid username or password'. (Test: TC\_01 covers invalid login; TC\_02 covers empty fields.) |
| FR-3: User Logout | The system shall allow a logged-in user to log out. | Upon logout, the user’s session is terminated and the user is redirected to the login page. (Test: TC\_08.) |
| FR-4: Navigation Bar | The system shall display a persistent navigation bar on all authenticated pages, containing links to Portfolio, Search, Strategies, History, and Logout. | Navigation bar must be visible and clickable on all authenticated pages. All links navigate correctly. |
| FR-5: Portfolio Dashboard (Insights) | The system shall display a Portfolio Overview page with summary boxes, asset distribution, and performance charts. | The four summary values and holdings data must load without error (TC\_03, TC\_04). |
| FR-6: Holdings Table | The system shall list each stock holding with columns: Ticker, Shares, Current Price, Total Value, Unrealized P/L, and Action button. | Holdings table displays correctly for the user and allows scrolling if needed. |
| FR-7: Trade Dialog | Clicking 'Trade' button shall open a dialog allowing user to buy or sell stock quantity. | Dialog opens correctly and allows input/submit. (Test: TC\_05). |
| FR-8: Manual Trading (Buy/Sell Stocks) | System allows manual buy/sell orders with validation of funds/shares. | After buy, balance reduces and holdings update (TC\_06). After sell, balance increases and holdings update (TC\_07). Errors shown for insufficient funds/shares. |
| FR-9: Stock Data Management | Maintain a stock database with metadata for at least 11,500 stock symbols. Updatable daily or on-demand. | Database stores accurate symbol/company/exchange data. |
| FR-10: Stock Search | Users can search stocks by symbol or name. | Search results appear in <2s with current market data. |
| FR-11: Live Stock Charting | Display live price charts with intraday/historical series. | Chart auto-refreshes and updates price trend correctly. |
| FR-12: Historical Price Data | Allow viewing historical price data with chart and table. | Chart and table must accurately reflect stored data. |
| FR-13: Technical Indicator Visualization | Offer indicators (e.g. MA, RSI, MACD) overlaid on charts. | Indicators correctly calculated and plotted. |
| FR-14: Strategy Definition | Allow user to define strategies with rules and risk parameters. | Users can create, save, edit, and delete strategies. Rules validated for correctness. |
| FR-15: Strategy-Based Auto-Trading | Execute trades based on user strategies at intervals or triggers. | Strategy triggers correctly place orders via Alpaca API and log results. |
| FR-16: Trade Execution & Confirmation | Handle all orders via Alpaca with confirmation. | Order updates balance and holdings. Rejections handled with messages. |
| FR-17: Email Notifications | Send email notifications for events such as automated trades. | Automated trades send correct email to user’s address. |
| FR-18: Price Forecasting (ARIMA) | Provide ARIMA-based price forecasts for stocks. | Forecast results displayed within reasonable time on prediction page. |
| FR-19: Stock Recommendations (Clustering) | Provide recommendations via K-Means clustering. | Recommendations displayed with categories (Strong Buy…Strong Sell). Users can filter. |
| FR-20: Trade History & Reporting | Maintain log of all trades and performance charts. | History table and charts accurately display past trades and P/L. |
| FR-21: Portfolio Management | Allow users to set risk parameters, alerts, and watchlists. | System enforces risk limits and saves user settings. |

# 

# 6. Non-Functional Requirements

Non-functional requirements describe how the system performs under various conditions. They set quality attributes and constraints on functional features. Key non-functional requirements for TradeAdvance include:

| NFR ID | Description |
| --- | --- |
| NFR-1: Performance | The system shall respond to user actions quickly. Page loads (dashboard, search results, charts) should complete within 2 seconds under normal load. It shall handle at least 100 concurrent users without performance degradation. Response to a stock search query must return results within 1–2 seconds. |
| NFR-2: Scalability | The architecture should allow scaling (e.g. database, API capacity) to support increasing data and users. It should be able to scale vertically or horizontally (e.g. more server instances) without redesign. |
| NFR-3: Availability and Reliability | The platform shall be available to users 99.9% of trading hours. Scheduled maintenance windows shall be announced in advance. In case of failures (e.g. Alpaca outage), the system must fail gracefully (show a maintenance message) rather than crash. The system shall maintain data integrity (e.g. transactions are ACID-consistent). |
| NFR-4: Security | All data transmissions must use secure protocols (HTTPS with SSL/TLS). User passwords shall be stored hashed (e.g. bcrypt). The system shall enforce authentication and role-based access. Sensitive data (API keys, personal information) must be encrypted at rest and in transit. Regular security measures (input validation, protection against common web attacks) shall be implemented. The system must comply with relevant data protection standards (e.g. GDPR for EU users, or industry best practices). |
| NFR-5: Usability | The user interface should be intuitive for novice traders. The system shall follow common UI/UX best practices (consistent layout, clear labels, help tooltips). Usability testing should ensure that first-time users can perform key tasks (e.g. placing a trade) without confusion. |
| NFR-6: Compatibility | The web interface must support modern browsers (latest Chrome, Firefox, Safari, Edge) on desktop and mobile. It should display correctly on different screen resolutions. Any third-party libraries must be compatible with the platform. |
| NFR-7: Maintainability and Extensibility | The codebase should be modular and documented to facilitate updates (e.g. adding new features or models). APIs and data schemas should be versioned so that extensions (new indicators, new markets) can be added without breaking existing features. |
| NFR-8: Regulatory Compliance | As a trading platform, the system should adhere to financial regulations. For example, it must maintain audit logs of all trades and user actions. It should ensure user data privacy and comply with laws (SEC, FINRA guidelines or GDPR/CCPA as applicable). Detailed compliance measures (KYC/AML) are beyond MVP scope but should be noted for future development. |

# ****7. User Interface Requirements****

The platform’s user interface shall be web-based, responsive, and intuitive for retail traders. The UI must support both desktop and mobile browsers with consistent design language across all modules.

### ****7.1 General UI Requirements****

* A **top navigation bar** shall be present on all authenticated pages, with links to all modules (Portfolio Insights, Stocks, Strategies, Ordered by Algorithm, Trade Manually, Recommendations, Predict Future, History).
* All **numeric and financial data** (balance, P/L, prices) must be clearly labeled and displayed with appropriate formatting (e.g., currency symbols, +/- signs).
* **Error messages** (e.g., “Invalid input”, “Insufficient funds”) must be displayed in **red** and be descriptive.
* **Pop-up dialogs** (e.g., Trade confirmation) must include **“Confirm”** and **“Cancel”** actions.
* **Forms** must validate inputs before submission (numeric checks, required fields, dropdown constraints).
* Consistent **branding, fonts, and color palette** shall be applied across the application.

### ****7.2 UI Pages & Figures****

Each major page will include placeholder figures (to be replaced by actual designs).

* **Figure 1.1: Portfolio Insights Dashboard**  
  Layout includes:
  + Summary statistics (Cash Balance, Total Portfolio Value, Total Investment, Current Market Value).
  + Holdings table (Ticker, Company, Shares, Avg Cost, Invested, P/L, Net Change, Action).
  + P/L chart and portfolio distribution pie charts.
* **Figure 1.2: Stock Search and List**
  + Searchable list of 11,500+ stocks with real-time prices.
  + Filters for RSI, SMA, new highs/lows, etc.
* **Figure 1.3: Stock Detail and Live Chart**
  + Individual stock page with live chart.
  + Toggleable indicators (SMA, RSI, Bollinger Bands, MACD).
  + Historical data table (OHLCV).
* **Figure 1.4: Strategy Management**
  + List of user-defined strategies.
  + Drill-down to assets included in each strategy.
* **Figure 1.5: Automated Orders Page**
  + Table of orders generated by algorithms.
  + Email notification mockup included.
* **Figure 1.6: Manual Trading Page**
  + Buy/Sell order form with ticker, quantity, price, and confirmation dialog.
* **Figure 1.7: Recommendations Page**
  + Long-Term (clustering results: Strong Buy → Strong Sell).
  + Short-Term (indicator + signal selection).
* **Figure 1.8: Price Prediction Page**
  + Predicted stock price with ARIMA forecast.
  + Graph visualization of actual vs. predicted.
* **Figure 1.9: Portfolio History Page**
  + Transaction list (buy/sell, quantity, price, realized P/L).
  + Historical performance chart.

### ****7.3 Usability & Accessibility****

* **Consistency**: Navigation and UI layout must remain consistent across all modules.
* **Accessibility**: Labels must be screen-reader friendly; sufficient color contrast must be maintained.
* **Error Prevention**: Default values, dropdowns, and constrained inputs should be used to minimize user errors.
* **Responsive Design**: Pages must adapt to various screen resolutions (desktop, tablet, mobile).
* **User Guidance**: Tooltips or inline hints should be provided where calculations or indicators are technical.

# 8. Data Requirements

TradeAdvance manages financial and user data. Key data entities and structures include:

| Entity | Description |
| --- | --- |
| User Account | Stores user ID, username, email, hashed password, registration date, and optional profile settings (e.g. risk preferences). |
| Portfolio/Holdings | For each user, a portfolio record containing holdings. A Holding record includes: user ID, stock ticker, quantity owned, average buy price. The portfolio also tracks cash balance and total value. |
| Stock Data | A Stock table: symbol (primary key), company name, exchange, and cached last price. Historical price data may be stored in a PriceHistory table or fetched on-demand from Alpaca. |
| Trade Transaction | Each trade generates a Transaction record: user ID, datetime, type (buy/sell), ticker, quantity, price per share, total value, fees (if any), and realized profit/loss. Used for history and performance charts. |
| Strategy | A Strategy table storing user-defined strategies: strategy ID, user ID, name, parameters (indicator names, thresholds), schedule (e.g. daily at market open). |
| Recommendations/Forecasts | Results of clustering and ARIMA may be transient or stored. Tables may include Recommendation (user ID, stock, category) and Forecast (user ID, stock, date, predicted\_price). |

## Data Integrity Constraints

• Usernames and emails must be unique.

• Holdings quantities must be non-negative (no short selling in MVP).

• Trade records must reconcile correctly (buy adds to holdings, sell reduces holdings).

• Historical data should be normalized (one record per stock-date combination).

## Data Backup

The system shall perform periodic backups of the database to prevent data loss. On restart, the system should recover the last known good state.

# 9. External Interfaces

TradeAdvance interacts with external systems as follows:

| Interface | Description |
| --- | --- |
| Market Data API | Provides real-time and historical stock data and order execution via Alpaca’s REST/streaming APIs. Must securely handle API keys. Must follow Alpaca’s protocol with error handling for API failures. |
| Trading API | Handles trade execution. Sends orders via Alpaca and listens for confirmations. Transactional: on confirmation update local DB; on error notify user. |
| Email/Notification Service | Connects to email service (SMTP or third-party) to send notifications (password reset, trade alerts). Must handle failures (retry/log errors). |
| Frontend–Backend API | Frontend communicates with backend via RESTful JSON APIs (/api/login, /api/register, /api/portfolio, /api/trade, /api/strategy, /api/recommendations). Inputs validated. JSON schemas defined. |
| User Interface | System accessible via web browsers (desktop/mobile). No native mobile app included in MVP. UI consumes backend APIs and renders data correctly. |
| System Administrator Interface | (Optional) Admin console or DB access for system administrators to manage data. Not exposed to end users. |

# 10. Compliance and Quality Attributes

## Audit Trail

The system shall keep detailed logs of user activities and system events. All trades (executed orders) must be logged in an immutable audit trail for compliance. These logs are part of the transaction history.

## Data Privacy

The system shall comply with privacy regulations. It will not sell or share user personal data. Personally identifiable information (PII) such as email addresses must be stored securely.

## Financial Compliance

Any trading platform must adhere to financial regulations. TradeAdvance will follow good practice such as not allowing trades outside market hours, maintaining sufficient funds check, and respecting the user’s margin and credit limits. While a full regulatory compliance (e.g. SEC registration) is beyond this scope, the system design must allow future implementation of compliance controls (e.g. approval workflows).

## Intellectual Property

All software libraries and images used must have appropriate licenses.

## Accessibility

The UI should adhere to basic accessibility standards (e.g. screen-reader friendly labels, contrast for text).

# 11. Use Cases

The following are representative use cases for TradeAdvance:

## Use Case 1: User Registration and Login

Actor: New User  
Description: A user navigates to the registration page, enters valid credentials, and submits. The system creates an account. The user then logs in successfully.  
FRs Covered: FR-1, FR-2, FR-3.

## Use Case 2: View Portfolio Dashboard

Actor: Trader  
Description: After login, the user views the Portfolio page. The system displays the summary boxes, holdings table, and performance chart (Figure 1.1).  
FRs Covered: FR-4, FR-5, FR-6, FR-8 (viewing holdings).

## Use Case 3: Search and Analyze Stock

Actor: Trader  
Description: The user uses the search bar to find 'AAPL'. The system returns Apple’s stock details. The user selects AAPL and views its live chart and historical data.  
FRs Covered: FR-10, FR-11, FR-12, FR-13.

## Use Case 4: Place a Manual Trade

Actor: Trader  
Description: On the AAPL page or portfolio, the user chooses to Buy 10 shares. The trade dialog opens (FR-7); the user enters 10 and submits. The system executes the order via Alpaca (FR-8) and updates the portfolio.  
FRs Covered: FR-7, FR-8, FR-16.

## Use Case 5: Create and Run Strategy

Actor: Trader  
Description: The user creates a new strategy (e.g. momentum strategy) with specific rules. On market open, the system evaluates the strategy and executes any triggered trades (FR-15). The user is notified and can review the updated holdings.  
FRs Covered: FR-14, FR-15, FR-16, FR-17.

## Use Case 6: View Recommendations and Forecast

Actor: Trader  
Description: The user goes to the Recommendations page. The system displays long-term recommended stocks (from clustering) and short-term signals (FR-19) (Figure 1.7), and a list of ARIMA-based forecasts for selected tickers (FR-18) (Figure 1.8). The user can filter by 'Buy' or 'Sell' signals.  
FRs Covered: FR-18, FR-19.

## Use Case 7: View Trade History

Actor: Trader  
Description: The user accesses the History page. The system shows a table of all past trades and a profit/loss chart (Figure 1.9).  
FRs Covered: FR-20.

# 12. Risk and Mitigation

Key project risks include:

## API Dependence

Reliance on Alpaca API. Mitigation: Use dummy data or backup data sources if Alpaca is down; design with abstraction so API keys/config can be changed.

## Data Accuracy

Inaccurate market data could mislead users. Mitigation: Cross-verify critical data; use official sources.

## Security Breach

Unauthorized access or data leak. Mitigation: Enforce strong authentication, use HTTPS, regularly update dependencies, perform security audits.

## Model Inaccuracy

Forecasts or recommendations may be wrong, leading to user dissatisfaction. Mitigation: Clearly label them as predictions; allow users to disable/ignore them; improve models iteratively.

## Performance Bottlenecks

Slow response with growing data. Mitigation: Profile and optimize queries; index database tables; use caching for frequent requests.

## Regulatory Changes

New trading regulations. Mitigation: Design system flexible to incorporate compliance rules; consult legal expertise.

# 13. Glossary and References

## ARIMA

Autoregressive Integrated Moving Average, a time-series forecasting model.

## K-Means Clustering

Unsupervised ML algorithm for grouping similar items.

## Portfolio

The collection of a user’s holdings and cash balance.

## TradeAdvance

The platform being developed (as described in the project report).