Technical Documentation

Project: MarketSphere - The 360° View of Investing

Table of Contents

1. Project Overview

MarketSphere is a comprehensive financial web application designed to bridge the gap between beginner education and active market tracking. It serves as a dual-purpose platform: a learning hub ("Stock School") for new investors and a market dashboard for tracking assets

The application aggregates financial data through a hybrid approach:

1. **Real-Time Data:** Integrates with the Alpha Vantage API to provide live, up-to-the-minute stock quotes for the US market.
2. **Dynamic Data**: Uses a custom web scraper (cheerio) to fetch the latest IPO listings and subscription statuses from InvestorZone, caching them to ensure performance.
3. **Static Data**: Delivers curated, reliable information on Indian brokers, mutual funds, and market sectors via local JSON databases.

The project is built as a Responsive Web Application (PWA-ready) that offers a seamless experience across desktop and mobile devices, featuring a persistent dark mode and offline capabilities.

Core App Features

- **Real-Time Market Dashboard:**
    - Provides a live search interface for US Stocks (e.g., AAPL, MSFT).
    - Displays real-time price, percentage change, volume, and day's high/low.
    - Visual Feedback: Color-coded indicators (Green/Red) for positive or negative market movement.
- **Latest IPO Tracking:**
    - Ongoing IPOs: Dynamically scraped data showing current price bands and issue sizes.

    - Status Badges: Visual labels for "LIVE NOW" or "UPCOMING" to guide investor attention.
    - Listed Performance: Tracks listing gains/losses for recently listed companies.
- **Broker Comparison Engine:**
    - Compares 17 top Indian brokers (Zerodha, Upstox, etc.) side-by-side.
    - Detailed Metrics: Breakdowns of account opening fees, maintenance charges, and brokerage rates.
    - Infinite Scroll: Optimized list view that loads brokers progressively for better performance.
- **Stock School (Education Hub):**
    - A structured curriculum with 4 learning levels (Beginner to Practical).
    - Contains 16 modules covering topics from "What is a Stock?" to "Technical Analysis".
- **Sector & Fund Analysis:**
    - Sectors: Visual grid of 8 major market sectors (Auto, IT, Pharma) with market cap distribution.
    - Mutual Funds: Curated list of top-performing funds with 1Y and 3Y return history.
- **Dark Mode & UI**
    - Full site-wide theme toggle that persists using localStorage.
    - Fully responsive, mobile-first design for all pages.

Core Technical Features

- **Hybrid Data Architecture:**
    - Seamlessly merges Live API data (Stocks), Scraped data (IPOs), and Static JSON (Brokers/Education) into a unified REST API.
- **Intelligent Caching System:**
    - Implements server-side caching (TTL: 1 hour) for IPO scraping to reduce latency and avoid IP bans from the source site.
- **Robust REST API (v1):**
    - Standardized endpoints (e.g., /api/v1/stock/search) with consistent JSON response structures.
    - Includes validation middleware to prevent malformed requests (e.g., invalid stock symbols).
- **Global Error Handling System:**
    - Backend: Centralized error responses (400, 404, 500) for API failures.
    - Frontend: A "Toast" notification system (Success/Error/Warning) that provides instant visual feedback to users.
- **Progressive Web App (PWA) Features:**
    - Service Worker: Caches core assets (CSS, JS, Home Page) to allow the app to load

faster and function partially offline.

    - Theme Persistence: Remembers user preference for Dark/Light mode across sessions using localStorage.

- **Automated Integration Testing:**
    - A comprehensive test suite verifies that all API endpoints return the correct data types, handle errors gracefully, and respond within acceptable time limits.

---

2. Technology Stack

This project utilizes the following technologies:

- **Backend:**
    - **Node.js:** JavaScript runtime environment.
    - **Express.js:** Web application framework for building the REST API.
- **Frontend:**
    - **HTML5:** Standard markup for web pages.
    - **CSS3:** Custom styling, including Flexbox, Grid, and Dark Mode.
    - **Vanilla JavaScript (ES6+):** Client-side logic, DOM manipulation, and API fetching (fetch).
- **API & Data:**
    - **REST API:** Custom API built with Express.js.
    - **JSON:** Used as the format for static data storage (in public/data/) and API responses.
- **Testing:**
    - **Jest:** Testing framework for running tests.
    - **Supertest:** HTTP assertion library for testing API endpoints.
- **Web Scraping:**
    - **Axios:** Promise-based HTTP client for making requests to external sites.
    - **Cheerio:** Server-side library for parsing and manipulating HTML (used in ipoService.js).
- **External APIs:**
    - **Alpha Vantage:** Used for the real-time US Stock Search feature.
- **Security:**
    - **Helmet**, **CORS**, **Rate Limit:** HTTP headers, cross-origin resource sharing, and DDOS protection.

---

3. Project Structure (Blueprint)

The project is organized into a standard Express.js backend (index.js, services/) and a static frontend (public/).

```
/ (Your Main Project Folder)
 ├── index.js              # Entry point: Express App, API Routes, & Server Config
 ├── package.json          # Project metadata & dependencies
 ├── package-lock.json     # Dependency version lock file
 ├── .env                  # Environment variables (PORT, API Keys)
 ├── .gitignore            # Files to ignore in version control
 |
 ├── __tests__/            # Test Suite
 |   └── integration.test.js   # API integration tests using Supertest
 |
 ├── services/             # Business Logic Layer
 |   └── ipoService.js     # Web scraper & caching logic (InvestorZone source)
 |
 ├── public/               # Frontend Static Assets (Served via Express)
 |   ├── index.html        # Landing Page
 |   ├── ipo.html          # IPO Listing Page
 |   ├── market.html       # Live Stock Search Page
 |   ├── brokers.html      # Broker Comparison Page
 |   ├── mutual-funds.html     # Mutual Funds Page
 |   ├── sectors.html      # Sector Analysis Page
 |   ├── stock-school.html     # Education Hub (Levels 1-4)
 |   ├── module-1.html     # Individual Learning Module
 |   ├── signin.html       # Authentication UI
 |   ├── sw.js             # Service Worker (PWA/Offline Caching)
 |   ├── style.css         # Global Styles (Dark mode, Responsive Grid)
 |   |
 |   ├── data/             # Static Database (JSON)
 |   |   ├── brokers.json      # Broker details (Zerodha, Upstox, etc.)
 |   |   ├── funds.json        # Mutual fund schemes
 |   |   ├── ipos.json         # Backup/Static IPO data
 |   |   ├── module-1.json     # Educational content
 |   |   ├── sectors.json      # Market sector info
```

```
|   |   └── stock-school.json # Curriculum structure
|   |
|   └── js/                 # Client-Side Logic
|       ├── home.js         # Toast notifications & login checks
|       ├── ipo.js          # Fetches & renders IPOs (Ongoing/Upcoming/Listed)
|       ├── market.js       # Stock search logic & validation
|       ├── brokers.js      # Broker list rendering & infinite scroll
|       ├── sectors.js      # Sector grid rendering
|       ├── mutual-funds.js  # Fund card rendering
|       ├── stock-school.js  # Educational module rendering
|       ├── module-1.js      # Dynamic content loader for lessons
|       ├── theme-toggle.js  # Dark/Light mode switcher
|       └── error-handler.js  # Global error handling & toast system
|
└── node_modules/           # Installed dependencies
```

---

4. REST API Documentation (v2.0.0)

Base URL: http://localhost:3000/api/v1

4.1 Rate Limiting

Limit: 100 requests per minute per IP address

4.2 Response Format

Success Response

```
{
  "success": true,
  "data": [...],
  "pagination": {
    "total": 50,
    "page": 1,
    "limit": 10,
    "totalPages": 5
  }
}
```

Error Response

```
{
```

```
  "success": false,
  "error": "Error message here",
  "details": [...]  // Optional validation details
}
```

4.3 Endpoints

IPO API

- **Get IPOs by Status:** Retrieve IPOs filtered by status. Data is scraped live from InvestorZone, with a fallback to ipos.json on failure.
    - **Endpoint:** GET /api/v1/ipos
    - **Query Parameters:**
        - status (string): Filter: ongoing, upcoming, listed, all. Default: all.
    - **Example:** curl "http://localhost:3000/api/v1/ipos?status=ongoing"

Brokers API

- **Get All Brokers:** Retrieve broker information from brokers.json with filtering.
    - **Endpoint:** GET /api/v1/brokers
    - **Query Parameters:**
        - rating (float): Minimum rating (0-5).
    - **Example:** curl "http://localhost:3000/api/v1/brokers?rating=4.0"

Search IPOs Search IPOs by name and price range.

- **Endpoint:** GET /api/v1/ipos/search
- **Query Parameters:**
    - q (string): Search query (min: 2 chars).
    - minPrice (integer): Minimum price filter.
    - maxPrice (integer): Maximum price filter.

Get Single IPO Retrieve detailed information about a specific IPO by its URL-friendly name.

- **Endpoint:** GET /api/v1/ipos/:name
- **Example Request:**

  curl "http://localhost:3000/api/v1/ipos/medi-assist-healthcare"

Mutual Funds API

Get All Mutual Funds Retrieve mutual fund information from funds.json with optional category filtering.

- **Endpoint:** GET /api/v1/funds
- **Query Parameters:**
    - category (string): Filter by category (partial match).
- **Example Request:**

  curl "http://localhost:3000/api/v1/funds?category=equity"

Sectors API

Get All Sectors Retrieve sector information from sectors.json.

- **Endpoint:** GET /api/v1/sectors
- **Example Request:**

  curl "http://localhost:3000/api/v1/sectors"

Stock School API

Get All Modules Retrieve educational modules from stock-school.json organized by level.

- **Endpoint:** GET /api/v1/stock-school
- **Example Request:**

  curl "http://localhost:3000/api/v1/stock-school"

Stock Search API (Real-Time)

Search US Stocks (External API) Search for real-time US stock quotes using the Alpha Vantage API.

- **Endpoint:** GET /api/v1/stock/search
- **Query Parameters:**
    - symbol (string, **Required**): US stock symbol (1-10 chars, alphanumeric).
- **Example Request:**

  curl "http://localhost:3000/api/v1/stock/search?symbol=AAPL"
- **Example Success Response:**


```
{
  "success": true,
  "data": {
    "symbol": "AAPL",
    "price": 182.52,
    "change": 1.23,
    "changePercent": 0.68,
    "open": 181.29,
```

```
    "high": 183.45,
    "low": 180.88,
    "volume": 52483920,
    "previousClose": 181.29
  }
}
```


---

5. Error Codes Reference Guide

This section provides a reference for all error codes, messages, and troubleshooting steps.

5.1 HTTP Status Codes

Code	Status	Description
200	OK	Request succeeded.
301	Moved Permanently	Redirect to new v1 endpoint.
400	Bad Request	Invalid parameters or validation failure.
403	Forbidden	CORS violation.
404	Not Found	Resource or stock symbol doesn't exist.
429	Too Many Requests	Rate limit exceeded (>100 req/min).
500	Internal Server Error	Unexpected server error.
504	Gateway Timeout	External API (Alpha Vantage) timeout.

5.2 API Error Codes

E001 - Validation Errors

Code: VALIDATION_FAILED

Message: "Validation failed"

HTTP Status: 400

Details:


```
{
  "success": false,
  "error": "Validation failed",
  "details": [
    {
      "field": "symbol",
      "message": "Symbol must contain only letters and numbers"
```

```
    }
  ]
}
```

Cause: Missing or incorrectly formatted query parameters (e.g., symbol=AAPL@).

E002 - Rate Limit Exceeded

- **Code:** RATE_LIMIT_EXCEEDED
- **Message:** "Too many requests. Please try again in 1 minute."
- **HTTP Status:** 429
- **Response:**

```
{
  "success": false,
  "error": "Too many requests. Please try again in 1 minute.",
  "retryAfter": 45
}
```

E003 - Resource Not Found

- **Code:** RESOURCE_NOT_FOUND
- **Message:** "Stock \"INVALID\" not found"
- **HTTP Status:** 404
- **Cause:** A validly formatted but non-existent stock symbol was provided.

E004 - External API Failure

- **Code:** EXTERNAL_API_ERROR
- **Message:** "Failed to fetch stock data" or "Request timeout. Please try again."
- **HTTP Status:** 500 or 504
- **Cause:** The Alpha Vantage API is down, or the request timed out (10s).

---

6. Testing Overview

Framework: Jest & Supertest **File:** __tests__/integration.test.js

- **Integration Testing:** Tests the *actual* API endpoints and server logic (not just isolated functions).
- **Mocking:** Uses jest.mock('axios') to simulate Alpha Vantage API responses, ensuring tests don't consume real API credits or rely on external network stability.
- **Coverage:**
    - **Static Endpoints:** Verifies brokers, funds, sectors return 200 OK and correct data structure.
    - **Stock Search:** Tests valid symbols, missing symbols (400), invalid formats (400), and API timeouts (504).

- **Redirects:** Ensures backward compatibility routes (e.g., /api/ipos) redirect correctly.

---

7. Key Features & Highlights

1. **Hybrid Data Model:** Combines **Static JSON** (for speed/reliability on constant data like brokers) with **Live Scraping** (for volatile data like IPOs) and **External APIs** (for real-time stock prices).
2. **Caching Strategy:** The ipoService.js implements node-cache (TTL: 3600 seconds/1 hour) to minimize scraping overhead and prevent getting blocked by the target site.
3. **Security Best Practices:**
    - **Helmet:** Adds secure HTTP headers.
    - **Rate Limiting:** Protects against brute-force/DDOS.
    - **Environment Variables:** Sensitive keys (ALPHA_VANTAGE_API_KEY) are stored in .env.
4. **Modern Frontend:**
    - **Dark Mode:** Persisted via localStorage in theme-toggle.js.
    - **PWA Ready:** Includes sw.js (Service Worker) for basic offline caching of assets.
    - **Infinite Scroll:** Implemented in brokers.js for better performance with large lists.

---

8. Future Enhancements

The following features are planned for future releases:

- **User Watchlists:** Building protected API endpoints for users to create, read, update, and delete their personal stock watchlists.
- **Portfolio Tracking:** Allowing users to track their holdings and performance.
- **Price Alerts (Push & Email):** Allow authenticated users to set price targets for stocks (e.g., "Alert me if AAPL goes below $180") and receive push notifications or emails.
- **Advanced Interactive Charting:** Integrate a full-featured charting library (like TradingView or Highcharts) to replace the simple price display on the market.html page.
- **Interactive Financial Calculators:** Build the actual tools for the "Fin Calculators" link in your footer, such as an SIP Calculator, Retirement Planner, or Loan EMI Calculator.
- **Deeper Fundamental Data:** Enhance the stock search to also return detailed company financials (e.g., P/E Ratio, EPS, Revenue, Income Statements).
- **Economic & Earnings Calendar:** Add a new section to show upcoming important events, such as company earnings reports, major economic announcements, or market holidays.

---

9. How to Run Your Project

Follow these steps to get the application running locally.

Prerequisites

- **Node.js** (v14 or higher)
- **npm** (Node Package Manager)

Step 1: Installation
Open your terminal in the project root folder and run:
Bash
npm install
This installs Express, Cheerio, Axios, Jest, and other dependencies defined in package.json.

Step 2: Environment Setup
You need to configure your API keys.

1. Open the .env file.
2. Add your Alpha Vantage API key (or keep the default if testing):

Code snippet
PORT=3000
ALPHA_VANTAGE_API_KEY=YOUR_REAL_API_KEY_HERE

Step 3: Run the Server
To start the application:
Bash
npm start

- The server will start at: http://localhost:3000
- Open this URL in your browser to view the app.

Step 4: Run Tests
To execute the integration test suite:
Bash
npm test