Product Requirements Document

Stock Broker Client Web Dashboard

Version: 1.0
Last Updated: December 7, 2025
Status: Draft

---

1. Product Overview

1.1 Purpose

A real-time web dashboard that allows stock broker clients to monitor subscribed stocks with live price updates without page refreshes.

1.2 Target Users

- Stock broker clients who need to monitor multiple stocks
- Individual investors tracking their portfolio
- Multiple concurrent users with different stock preferences

1.3 Key Value Proposition

- Real-time price monitoring without manual refresh
- Personalized stock watchlists
- Multi-user support with independent subscriptions

---

2. Objectives

2.1 Primary Goals

- Enable users to login and maintain session state
- Allow users to subscribe/unsubscribe to stocks dynamically
- Provide real-time price updates (every 1 second)
- Support multiple concurrent users with independent dashboards

2.2 Success Metrics

- User login success rate: >95%
- Price update latency: <1 second
- Dashboard uptime: >99%
- Support for 2+ concurrent users with separate subscriptions

---

3. User Stories

3.1 Authentication

As a stock broker client
I want to login using my email
So that I can access my personalized dashboard

Acceptance Criteria:

- User can enter email address
- System validates email format
- User session is maintained until logout
- Email is displayed in dashboard header

3.2 Stock Subscription

As a logged-in user
I want to subscribe to stocks from a predefined list
So that I can monitor only the stocks I care about

Acceptance Criteria:

- User can view list of 5 supported stocks: GOOG, TSLA, AMZN, META, NVDA
- User can add stocks to their watchlist
- User can remove stocks from their watchlist
- Subscription changes persist during session
- User sees only their subscribed stocks

3.3 Real-Time Price Updates

As a user monitoring stocks

I want to see prices update automatically
So that I don't have to manually refresh the page

Acceptance Criteria:

- Stock prices update every 1 second
- Updates happen without page reload
- Price changes are visually indicated (up/down)
- No lag or freezing during updates

3.4 Multi-User Support

As a user
I want my dashboard to be independent of other users
So that my subscriptions don't affect others

Acceptance Criteria:

- User A's subscriptions don't affect User B's view
- Both dashboards update asynchronously
- Each user maintains separate session state

---

4. Functional Requirements

4.1 Authentication Module

ID	Requirement	Priority
FR-1.1	System shall provide email input field	Must Have
FR-1.2	System shall validate email format	Must Have
FR-1.3	System shall maintain user session	Must Have
FR-1.4	System shall provide logout functionality	Must Have
FR-1.5	System shall display current user email	Should Have

4.2 Stock Subscription Module

ID	Requirement	Priority
FR-2.1	System shall support exactly 5 stocks: GOOG, TSLA, AMZN, META, NVDA	Must Have
FR-2.2	System shall allow users to add stocks to watchlist	Must Have
FR-2.3	System shall allow users to remove stocks from watchlist	Must Have

ID	Requirement	Priority
FR-2.4	System shall display only subscribed stocks	Must Have
FR-2.5	System shall show available stocks for subscription	Must Have

4.3 Price Update Module

ID	Requirement	Priority
FR-3.1	System shall generate mock stock prices	Must Have
FR-3.2	System shall update prices every 1 second	Must Have
FR-3.3	System shall update UI without page refresh	Must Have
FR-3.4	System shall indicate price direction (up/down)	Should Have
FR-3.5	System shall show percentage change	Should Have

4.4 Multi-User Module

ID	Requirement	Priority
FR-4.1	System shall support multiple concurrent sessions	Must Have
FR-4.2	System shall isolate user subscriptions	Must Have
FR-4.3	System shall update all active dashboards independently	Must Have

---

5. Technical Requirements

5.1 Frontend Technology Stack

- **Framework:** React.js (or Vue.js/Angular)
- **State Management:** React Hooks (useState, useEffect)
- **Styling:** Tailwind CSS or Material-UI
- **Build Tool:** Vite or Create React App

5.2 Data Management

- **Client-Side Storage:** Browser localStorage or sessionStorage
- **State Updates:** React state with interval-based updates
- **No Backend Required:** Mock data generation on frontend

5.3 Price Generation Algorithm

```
New Price = Current Price × (1 + (Random(-0.02, 0.02)))
Update Interval: 1000ms (1 second)
Random Range: ±2% maximum change per update
```

5.4 Browser Compatibility

- Chrome (latest 2 versions)
- Firefox (latest 2 versions)
- Safari (latest 2 versions)
- Edge (latest 2 versions)

---

6. Non-Functional Requirements

6.1 Performance

- Page load time: <2 seconds
- Price update processing: <100ms
- UI rendering: 60 FPS minimum

6.2 Usability

- Intuitive single-page interface
- Clear visual feedback for actions
- Responsive design for desktop (tablet/mobile optional)

6.3 Reliability

- No memory leaks from intervals
- Graceful handling of tab visibility changes
- Proper cleanup on component unmount

6.4 Scalability

- Support 2-10 concurrent users (browser sessions)
- Handle 5-20 subscribed stocks per user

---

7. User Interface Requirements

7.1 Login Screen

- Email input field with validation
- "Login" button
- Minimal, centered design

7.2 Dashboard Layout

Header:

- Application title
- Current user email
- Logout button

Main Area:

- Grid/list of subscribed stocks showing:
  - Stock ticker symbol
  - Current price
  - Price change indicator (↑/↓)
  - Percentage change
  - Unsubscribe button

Sidebar/Section:

- Available stocks list
- Subscribe buttons for each stock

7.3 Visual Indicators

- Green for price increases
- Red for price decreases
- Animations for price changes
- Loading states where appropriate

---

8. Data Model

8.1 User Object

```javascript
{
  email: "user@example.com",
  subscriptions: ["GOOG", "TSLA", "AMZN"]
}
```

8.2 Stock Price Object

```javascript
{
  ticker: "GOOG",
  price: 175.50,
  previousPrice: 174.80,
  change: 0.70,
  percentChange: 0.40,
  timestamp: 1701965432000
}
```

8.3 Supported Stocks

```javascript
["GOOG", "TSLA", "AMZN", "META", "NVDA"]
```

---

9. User Flow

9.1 First-Time User Flow

1. User opens application
2. User enters email address
3. User clicks "Login"
4. Dashboard displays with no subscriptions
5. User subscribes to desired stocks
6. Prices begin updating automatically

9.2 Returning User Flow (Same Session)

1. User opens application
2. Dashboard displays with previous subscriptions
3. Prices update automatically

9.3 Stock Management Flow

1. User views available stocks
2. User clicks "Subscribe" on desired stock
3. Stock appears in main dashboard
4. Prices update in real-time
5. User can click "Unsubscribe" to remove stock

---

10. Assumptions & Constraints

10.1 Assumptions

- Mock data is sufficient for demonstration
- No real-time API integration needed
- Single-device usage (no cross-device sync)
- Session data doesn't need persistence after browser close

10.2 Constraints

- Exactly 5 supported stocks (not configurable)
- Client-side only (no backend server)
- Price updates are simulated (not real market data)
- No historical price data or charts

10.3 Dependencies

- Modern web browser with JavaScript enabled
- Internet connection for initial load (if hosted)
- LocalStorage API support

---

11. Out of Scope (V1)

- Real stock market API integration
- Historical price charts
- Trading/buy/sell functionality
- Portfolio value calculation
- Price alerts/notifications
- User registration/password authentication
- Cross-device synchronization
- Mobile app version
- Dark mode
- Export data functionality

---

12. Future Enhancements (V2+)

12.1 Phase 2

- WebSocket integration for true real-time updates
- Historical price charts (candlestick/line)
- Price alerts when stocks hit thresholds
- Portfolio value tracking

12.2 Phase 3

- Real market data API integration
- User authentication with password
- Backend database for persistence
- Trading simulation functionality
- Mobile responsive design

12.3 Phase 4

- News feed integration

- Social features (share watchlists)
- Advanced analytics and insights
- Multi-currency support

---

13. Testing Requirements

13.1 Unit Testing

- Price generation algorithm
- Subscription add/remove logic
- Email validation

13.2 Integration Testing

- Multi-user scenarios
- Price updates across components
- Session management

13.3 User Acceptance Testing

- Login flow
- Subscribe/unsubscribe workflow
- Real-time updates verification
- Multi-tab testing (2+ users)

---

14. Deployment

14.1 Hosting Options

- Static hosting (Vercel, Netlify, GitHub Pages)
- No backend server required
- CDN for static assets

14.2 Environment

- Production environment only (no staging needed for MVP)

15. Timeline Estimate

Phase	Duration	Tasks
Design	2 days	UI mockups, component structure
Development	5 days	Core features implementation
Testing	2 days	Unit, integration, UAT
Deployment	1 day	Build and deploy
Total	**10 days**	

---

16. Risks & Mitigation

Risk	Impact	Probability	Mitigation
Memory leaks from intervals	High	Medium	Proper cleanup on unmount, interval management
Browser compatibility issues	Medium	Low	Test on multiple browsers, use polyfills
State synchronization bugs	High	Medium	Thorough testing of multi-user scenarios
Performance degradation	Medium	Low	Optimize re-renders, debounce updates

---

17. Approval

Role	Name	Signature	Date
Product Owner			
Tech Lead			
Stakeholder			

Document Control:

- Template Version: 1.0
- Next Review Date: Post-implementation
- Owner: Product Team