COLLEGE CODE : 9111

COLLEGE NAME : SRM Madurai College For Engineering And Technology

DEPARTMENT: Computer Science And Engineering

STUDENT NM-ID : 1AB1AA81B6B40D9B5C94D469B460D04A

1FFEB20A6540161A1F54E258167D1978

877B09181AE2E069AFF4C36AF16D1C4D

FF192F92083F64EB3754B0596244F018

ROLL NO ; 911123104040

911123104016

911123104032

911123104004

DATE : 29/09/2025

Completed the project named as phase 4

TECHNOLOGY PROJECT NAME : REAL TIME STOCK TICKER

SUBMITTED BY:

Rajesh Kumar M	mobile no : 8524925826
Harish manikandan R	mobile no : 6369119261
Nyson DM	mobile no : 8682054707
Arul kumaran M	mobile no : 8428853991

Additional Features

1. Multi-Asset Support

- Extend support beyond stocks to include **cryptocurrencies, forex, and commodities**.
- Allow users to track **diverse financial instruments** in one dashboard.

2. Personalized Notifications

- Push/email/SMS alerts when stocks hit user-defined thresholds.
- AI-driven **sentiment analysis** from news and social media for advanced alerts.

3. Historical Data & Insights

- Users can view **past performance trends** with customizable date ranges.
- Generate **insight reports** like top gainers/losers, sector-wise performance.

4. Portfolio Management

- Users can simulate or connect actual portfolios.
- Calculate **profit/loss, CAGR, volatility metrics**.

5. Social Trading Features

- Community watchlists and discussion threads.
- Ability to share trading ideas in-app.

UI/UX Improvements

1. Dashboard Redesign

- **Card-based layout** for quick stock summaries.
- Dark/light mode toggle.

2. Real-Time Charts

- Candlestick and volume charts with zoom/pan.
- Heatmaps for top 50 market movers.

3. Accessibility Enhancements

- Support for screen readers.
- Keyboard navigation for non-mouse users.

4. Personalization

- Drag-and-drop widget layout.
- Custom themes, fonts, and color schemes.

5. Mobile-First Approach

- Responsive design with React Native companion app.
- Native push notifications for alerts.

API Enhancements

1. Enhanced Endpoints

- `/stocks/{symbol}/history` → Returns historical prices.
- `/alerts/{userId}` → Fetch user-specific alerts.
- `/portfolio/{userId}` → Fetch holdings and P/L calculations.

2. API Gateway Integration

- Use **API Gateway (AWS or Kong)** to manage requests.
- Enable **rate-limiting** and request throttling for stability.

3. GraphQL Support

- Allow clients to fetch only required fields.
- Reduce payload size for mobile users.

4. Caching Layer

- Introduce **Redis caching** for frequently requested data.
- Store latest stock prices in-memory for fast retrieval.

API Schema Example (REST) :


```
{
  "symbol": "AAPL",
  "price": 175.40,
  "currency": "USD",
  "volume": 3400000,
  "timestamp": "2025-09-29T10:00:00Z"
}
```

Performance & Security Checks

1. Performance

- **Load Testing** with JMeter/k6.
- Optimize WebSocket server with horizontal scaling.
- Implement CDN for static content delivery.

2. Security

- Encrypt communication using **HTTPS + TLS 1.3**.
- Implement **JWT tokens** for user authentication.
- **SQL injection & XSS prevention** with sanitization libraries.
- Apply **OWASP Top 10 checks**.

3. Monitoring & Logging

- Real-time monitoring with **Prometheus + Grafana**.
- Error logging with **ELK stack (Elasticsearch, Logstash, Kibana)**.

Sample Security Code (JWT Middleware) :

```
function authenticateToken(req, res, next) {
  const token = req.headers["authorization"];
  if (!token) return res.sendStatus(401);
  jwt.verify(token, process.env.JWT_SECRET, (err, user) => {
    if (err) return res.sendStatus(403);
    req.user = user;
    next();
  });
}
```

Testing of Enhancements

1. Unit Testing

- Test new APIs (`/portfolio`, `/alerts`) with Jest.
- Mock external API calls using **nock**.

2. Integration Testing

- Verify WebSocket + Database sync.
- Test **portfolio updates → reflected in UI**.

3. Regression Testing

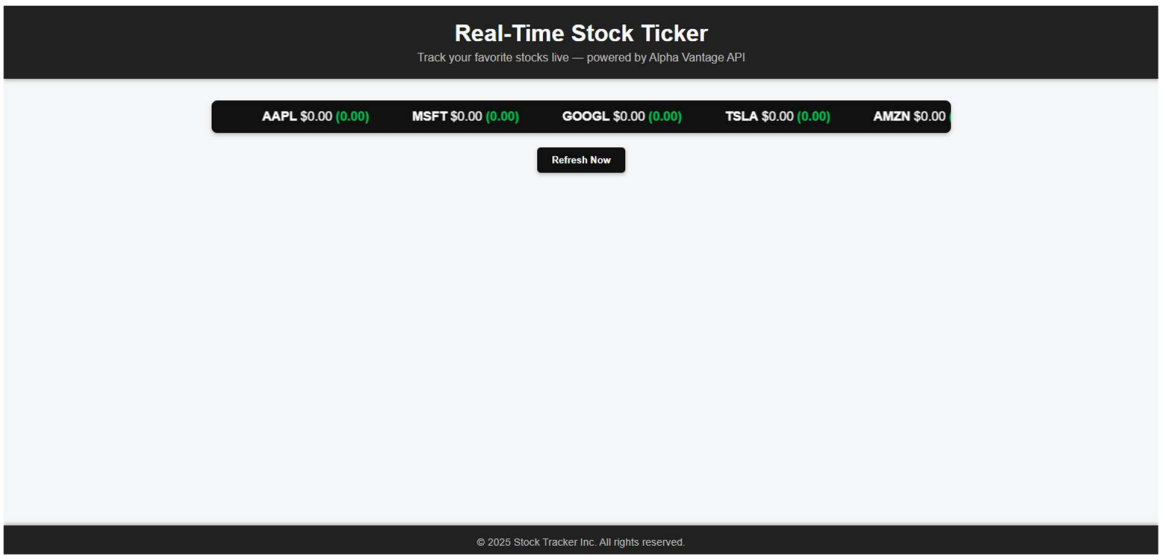
- Ensure new features don't break existing workflows.

4. User Acceptance Testing (UAT)

- Invite beta users to validate **UI improvements** and **new features**.
- Collect feedback via surveys & analytics.

5. Automation

- Set up CI/CD pipeline with **automated test suites**.
- Nightly builds trigger regression and load tests.



Deployment in Cloud Platform :

1. Cloud Providers

- **AWS**: EC2, RDS, Lambda, API Gateway.
- **Azure**: App Service + CosmosDB.
- **GCP**: Cloud Run + Firestore.

2. Deployment Architecture

Frontend (html, css)

Backend (Node.js) → EC2 / Kubernetes

Database (Sql)

Realtime Updates → WebSocket Service

Monitoring → CloudWatch/Grafana

3. CI/CD Pipeline

- GitHub Actions → Build & Test → Deploy to AWS/GCP.
- Canary releases for controlled rollouts.

4. Scaling

- Use **Kubernetes autoscaling** for high traffic.
- Separate **read & write DB replicas**.

5. Cost Optimization

- Spot instances for background jobs.
- Serverless Lambda for low-frequency alerts.

6. Final Outcome

- Highly scalable, secure, real-time system.
- Deployed on cloud with monitoring & rollback support.

7. Git hub repository link :

https://github.com/Arulkumaran17/Naan_mudhalvan.git