# **InvestIQ Deployment Manual**

## **Project Overview**

InvestIQ is an AI-powered investment assistant application built with Next.js. This guide focuses on deploying the frontend application located in the code/investiq folder of the repository.

## **System Architecture**

The application consists of three main components:

* **Frontend**: Next.js web application with Tailwind CSS (located in code/investiq)
* **Backend API**: Node.js/Express server for data processing
* **ML Services**: Python-based prediction services for market analysis

## **Frontend Structure**

The frontend code is organized as follows:

code/investiq/

├── app/

│ ├── components/ # Shared UI components

│ ├── dashboard/ # Dashboard page

│ ├── portfolio/ # Portfolio management page

│ ├── profile/ # User profile page

│ ├── recommendations/ # AI recommendations page

│ ├── stocks/ # Stock tracking page

│ ├── services/ # API service connectors

│ │ ├── DataService.ts

│ │ ├── NewsService.ts

│ │ └── StockService.ts

│ ├── globals.css # Global styles

│ └── layout.tsx # Root layout component

├── public/ # Static assets

├── config/ # Configuration files

├── next.config.js # Next.js configuration

└── package.json # Dependencies

## **Prerequisites**

### **Hardware Requirements**

* **Development**: 4-core CPU, 16GB RAM, 100GB storage
* **Production**:
  + Web servers: 2 vCPUs, 4GB RAM per instance

### **Software Requirements**

* Node.js 18.x or newer
* npm 9.x or newer
* Git

### **API Requirements**

* Finnhub API key
* AlphaVantage API key

## **Development Setup**

Clone the repository:  
  
 git clone https://github.com/htmw/2025S-Techno-Stack.git

cd 2025S-Techno-Stack/code/investiq

Install dependencies:  
  
 npm install

Configure environment variables:  
  
 # Create .env.local file

echo "NEXT\_PUBLIC\_API\_URL=http://localhost:8080" > .env.local

echo "NEXT\_PUBLIC\_FINNHUB\_KEY=your\_finnhub\_key" >> .env.local

Start the development server:  
  
 npm run dev

1. Access the application at http://localhost:3000

## **Production Deployment**

### **Build for Production**

Navigate to the frontend directory:  
  
 cd 2025S-Techno-Stack/code/investiq

Install dependencies:  
  
 npm install

Build the application:  
  
 npm run build

Start the production server:  
  
 npm start

### **Docker Deployment**

Create a Dockerfile in the frontend directory:  
  
 FROM node:18-alpine

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

RUN npm run build

EXPOSE 3000

CMD ["npm", "start"]

Build and run the Docker image:  
  
 docker build -t investiq-frontend:latest .

docker run -d -p 3000:3000 --env-file .env.production --name investiq-frontend investiq-frontend:latest

## **Configuration**

### **Environment Variables**

Configure the following environment variables in .env.local for development or .env.production for production:

* NEXT\_PUBLIC\_API\_URL: Backend API URL
* NEXT\_PUBLIC\_FINNHUB\_KEY: Finnhub API key
* NEXT\_PUBLIC\_ALPHAVANTAGE\_KEY: AlphaVantage API key

### **External Services Configuration**

The application connects to financial data APIs through service files:

1. StockService.ts: Handles stock data retrieval
2. NewsService.ts: Manages financial news feeds
3. DataService.ts: Provides unified data access layer

Update the API keys in config.ts to connect to these services.

## **Component Integration**

The InvestIQ frontend uses several key components:

1. **StockChart**: Displays stock price charts using Recharts
2. **HistoricalTrends**: Shows portfolio performance over time
3. **IPOCalendar**: Lists upcoming IPOs
4. **Sidebar**: Main navigation component
5. **DepositModal**: Handles user deposits

These components can be customized by modifying their respective files in the components directory.

## **Machine Learning Service Deployment**

The InvestIQ application leverages machine learning models for stock price prediction and sentiment analysis. The ML service should be set up as follows:

### **ML Service Structure**

code/investiq-ml/

├── app.py # Main Flask application

├── models/

│ ├── lstm/ # Time series prediction models

│ │ ├── lstm\_1d.h5 # 1-day prediction model

│ │ ├── lstm\_5d.h5 # 5-day prediction model

│ │ └── lstm\_30d.h5 # 30-day prediction model

│ └── sentiment/ # Sentiment analysis models

│ ├── gpt2/ # Fine-tuned GPT-2 model files

│ └── config.json # Model configuration

├── utils/

│ ├── preprocessing.py # Data preprocessing utilities

│ ├── prediction.py # Prediction endpoint utilities

│ └── sentiment.py # Sentiment analysis utilities

├── requirements.txt # Python dependencies

└── Dockerfile # Docker configuration for ML service

### **Prerequisites**

* Python 3.9 or newer
* pip package manager
* CUDA-compatible GPU for optimal performance (optional)
* Docker (for containerized deployment)

### **Development Setup**

Create and activate a Python virtual environment:  
  
 cd code/investiq-ml

python -m venv venv

source venv/bin/activate # On Windows: venv\Scripts\activate

Install dependencies:  
  
 pip install -r requirements.txt

Download or train ML models:  
  
 # Create model directories

mkdir -p models/lstm models/sentiment

# Download pre-trained models (example script)

python scripts/download\_models.py

Start the ML service:  
  
 python app.py

1. The service will be available at http://localhost:5000

### **ML Model Configuration**

#### **LSTM Stock Prediction**

The LSTM models require configuration in a JSON format:

{

"lstm": {

"input\_sequence\_length": 20,

"features": ["close", "volume", "high", "low", "open"],

"target": "close",

"normalization": "min\_max"

}

}

#### **GPT-2 Sentiment Analysis**

The GPT-2 model requires the following configuration:

{

"sentiment": {

"model\_type": "gpt2",

"model\_path": "models/sentiment/gpt2",

"max\_length": 512,

"use\_gpu": true

}

}

### **Docker Deployment**

Build the Docker image:  
  
 cd code/investiq-ml

docker build -t investiq-ml:latest .

Run the container:  
  
 docker run -d -p 5000:5000 --name investiq-ml investiq-ml:latest

For GPU support:  
  
 docker run -d -p 5000:5000 --gpus all --name investiq-ml investiq-ml:latest

### **API Endpoints**

The ML service exposes the following API endpoints:

**Stock Price Prediction**:  
  
 POST /api/predict/price

Request body:  
  
 {

"symbol": "AAPL",

"horizon": "5d",

"history": [

{"date": "2025-05-01", "close": 175.42, "volume": 78512345, "high": 176.82, "low": 173.95, "open": 174.10},

...

]

}

**Market Sentiment Analysis**:  
  
 POST /api/analyze/sentiment

Request body:  
  
 {

"text": "The company reported strong earnings, exceeding analyst expectations.",

"detailed": true

}

### **Integration with Frontend**

The ML service integrates with the frontend through the backend API. Update the following files to enable this integration:

Configure the ML service URL in the backend's .env file:  
  
 ML\_API\_URL=http://localhost:5000

1. Ensure the backend API has routes to proxy ML requests to avoid CORS issues.

## **Troubleshooting**

### **Common Issues**

1. **API connection errors**:  
   * Verify API keys in configuration
   * Check backend API status
   * Confirm correct API URL in environment variables
2. **UI rendering issues**:  
   * Clear browser cache
   * Check for JavaScript console errors
   * Verify Tailwind CSS compilation
3. **Build failures**:  
   * Ensure Node.js version compatibility
   * Check for dependency conflicts
   * Verify file permissions
4. **ML service issues**:  
   * Verify model files exist and are correctly placed
   * Check Python environment and dependency compatibility
   * For GPU acceleration, ensure CUDA is properly installed
   * Check system memory when loading large models