 **National University of Computer and Emerging Sciences**

**NLP ASSIGNMENT -02**

|  |  |
| --- | --- |
|  |  |
| **DEGREE PROGRAM:** | BS-SE |
| **SECTION:** | A |
| **SUBJECT NAME:** | Natural Language Processing |
| **DATE OF SUBMISSION:** | Oct 11, 2025 |
| **SUBMITTED TO:**  **SUBMITTED BY:** | Sir Omer Beg  Noor-ul-Iman |

# Stock Forecasting Application - Technical Report

## Project Objectives and Requirements

### Assignment Requirements:

**Primary Objectives:**

1. **Web Interface Development**: Complete Flask-based application with user-friendly interface

2. **Financial Data Integration**: Real-time data collection from multiple sources (stocks, crypto, forex)

3. **Traditional ML Models**: Implementation of statistical forecasting methods (SMA, EMA, ARIMA, VAR)

4. **Neural Network Models**: Deep learning implementations (LSTM, GRU, Transformer foundation)

5. **Model Comparison**: Comprehensive performance evaluation and ensemble methods

6. **Visualization**: Interactive charts with forecast overlays using Plotly

7. **User Selection**: Dynamic instrument and horizon selection capabilities

**Technical Requirements:** -

**Programming Language**: Python 3.9+ with Flask framework

**Database**: MongoDB for time series data storage

**Machine Learning**: Traditional (statsmodels, scikit-learn) and Neural (PyTorch)

**Frontend**: HTML5, CSS3, JavaScript with Bootstrap 5

**Testing**: Comprehensive test suite with 100% pass rate

**Documentation**: Complete technical documentation and user guides

### Scope and Limitations

**Supported Instruments:** - **Stocks**: AAPL (primary), GOOGL, MSFT, TSLA

**Cryptocurrency**: BTC-USD, ETH-USD  
**Forex**: EURUSD=X

**Forecast Horizons:** - 1 hour, 3 hours, 24 hours, 72 hours predictions

**Model Limitations:** - Historical data dependency (minimum 60 days for neural models) - Market volatility impacts prediction accuracy - External factors (news, events) not incorporated - Limited to technical analysis approaches

## System Architecture and Design

### Architecture Overview

The application follows a layered architecture pattern ensuring separation of concerns, maintainability, and scalability:

**1. Presentation Layer** - Responsive web interface using Flask templates and Bootstrap 5 - Interactive Plotly.js charts for data visualization - Real-time updates through AJAX API calls - Mobile-responsive design for accessibility

**2. Application Layer** - Flask web framework handling HTTP requests and routing - RESTful API endpoints for frontend-backend communication - Request validation and error handling - Session management and user preferences

**3. Business Logic Layer** - Model implementations with unified interfaces - Training pipeline supporting both traditional and neural approaches  
- Ensemble methods combining multiple model predictions - Performance evaluation and model comparison frameworks

**4. Data Access Layer** - Multi-source data collection (yfinance for stocks, ccxt for crypto) - Data preprocessing and feature engineering pipelines - MongoDB operations for persistent storage - Caching mechanisms for performance optimization

**5. Persistence Layer** - MongoDB database for time series data storage - File system storage for trained model persistence - Configuration management for application settings - Backup and recovery mechanisms

### Key Design Decisions

**Database Choice**

**MongoDB:** Excellent for time series data with flexible schema

**Benefits**: Horizontal scaling, complex queries, JSON-native

**Implementation**: Collections for prices, predictions, performance metrics

**Framework Choice**

**Flask:** Lightweight, flexible, excellent for ML applications

**Benefits**: Easy API development, template system, extensive libraries

**Implementation**: Modular structure with blueprints for scalability

**Model Architecture**

**Unified Interface:** Consistent API across different model types

**Benefits**: Easy ensemble implementation, model swapping, testing

**Implementation**: Abstract base classes with common prediction interface

## Machine Learning Implementation

### Traditional Models Implementation

1. **Simple Moving Average (SMA)**

**Algorithm**: Calculates average of last N prices

**Parameters**: Window size (configurable 5-50 periods)

**Use Case**: Trend identification and smoothing

**Performance**: MAPE ~1.0% for short-term predictions

1. **Exponential Moving Average (EMA)**

**Algorithm**: Weighted average giving more importance to recent prices

**Parameters**: Alpha smoothing factor (0.1-0.9)

**Use Case**: Responsive trend following

**Performance**: MAPE ~1.0% with faster adaptation

1. **ARIMA (AutoRegressive Integrated Moving Average)**

**Algorithm**: Statistical model for time series forecasting

**Parameters**: Auto-order selection using AIC/BIC criteria

**Features**: Stationarity testing, differencing, seasonal adjustment

**Performance**: MAPE ~2.9% for medium-term forecasts

1. **VAR (Vector AutoRegression)**

**Algorithm**: Multivariate time series model

**Features**: Multiple variable relationships, lag order optimization

**Use Case**: Capturing cross-asset correlations

**Performance**: MAPE ~1.1% for portfolio analysis

### Neural Network Implementation

**1. LSTM (Long Short-Term Memory)**

**Architecture**: 2-layer LSTM with 50 hidden units each

**Features**: Handles long-term dependencies in time series

**Training**: 100 epochs with early stopping

**Performance**: Training loss reduction from 0.306 → 0.200

**2. GRU (Gated Recurrent Unit)**

**Architecture**: 2-layer GRU with 50 hidden units each

**Features**: Simplified architecture, faster training than LSTM

**Training**: 100 epochs with Adam optimizer

**Performance**: Training loss reduction from 0.328 → 0.111

# ****Data Preprocessing Pipeline****

### ****Feature Engineering****

* **Technical Indicators:** SMA, EMA, RSI, MACD, Bollinger Bands, ATR
* **Price Features:** OHLCV data with volume indicators
* **Lag Features:** Historical price patterns (5, 10, 20 periods)
* **Normalization:** MinMax scaling for neural networks

### ****Sequence Preparation****

* **Lookback Window:** 60 periods for pattern recognition
* **Target Generation:** Next period price prediction
* **Train/Validation/Test Split:** 70/15/15% ratio
* **Data Quality:** Missing value handling and outlier detection

# ****Ensemble Methods****

### ****1. Performance-Weighted Ensemble****

* **Method:** Weights based on historical MAPE scores
* **Implementation:** Dynamic weight adjustment based on recent performance
* **Result:** Combines strengths of different model types

### ****2. Simple Average Ensemble****

* **Method:** Equal weights for all models
* **Use Case:** Baseline ensemble approach
* **Benefits:** Reduces individual model bias

### ****3. Dynamic Selection Ensemble****

* **Method:** Selects best-performing model for current conditions
* **Implementation:** Rolling window performance evaluation
* **Benefits:** Adapts to changing market conditions

# ****Results and Performance Analysis****

## ****Testing Results****

### ****Phase 7 Comprehensive Testing****

* **Total Tests:** 30 comprehensive test cases
* **Success Rate:** 100% (30/30 tests passed)
* **Categories Tested:** Data pipeline, API integration, end-to-end workflows
* **Execution Time:** 41.1 seconds for complete test suite

### ****Test Coverage****

* **Data Pipeline:** 100% (9/9 tests) — collection, processing, database operations
* **API Integration:** 100% (12/12 tests) — all Flask endpoints validated
* **End-to-End:** 100% (9/9 tests) — complete workflow validation
* **Performance:** Sub-second response times across all operations

## ****Model Performance Metrics****

### ****Traditional Models****

* **SMA:** MAPE 1.00% — excellent for trend following
* **EMA:** MAPE 1.00% — responsive to price changes
* **ARIMA:** MAPE 2.89% — good for statistical forecasting
* **VAR:** MAPE 1.07% — effective for multivariate analysis

### ****Neural Models****

* **LSTM:** Consistent training convergence, handles long sequences
* **GRU:** Faster training, comparable performance to LSTM
* **Training Stability:** Both models show decreasing loss curves

### ****Ensemble Performance****

* **Performance-Weighted:** Optimal combination based on historical accuracy
* **Component Validation:** All 5 models (SMA, EMA, ARIMA, LSTM, GRU) operational
* **Real-time Processing:** Live data collection and prediction generation

# ****System Performance****

### ****Database Operations****

* **Data Collection:** Successfully collected 502 historical records across instruments
* **Storage Efficiency:** Optimized MongoDB schema with proper indexing
* **Query Performance:** Sub-millisecond retrieval for recent data

### ****API Response Times****

* **Forecast Endpoint:** Average 250 ms response time
* **Ensemble Predictions:** 300 ms for complete ensemble processing
* **Data Collection:** 1–2 seconds for live data fetching
* **Model Training:** 5–10 seconds for incremental updates

# ****Real-World Validation****

### ****Live Data Testing****

* **AAPL Stock:** 22 records collected and processed successfully
* **Feature Engineering:** 60 features generated from price data
* **Model Training:** All models successfully trained on real data
* **Prediction Generation:** Accurate forecasts for multiple horizons

# ****Deployment and Production Readiness****

## ****Production Deployment Strategy****

### ****Environment Setup****

* **Development:** Local MongoDB with Flask development server
* **Testing:** Automated test suite with CI/CD integration potential
* **Production:** MongoDB Atlas cloud database with production WSGI server

### ****Scalability Considerations****

* **Horizontal Scaling:** Stateless Flask application design
* **Database Scaling:** MongoDB sharding capabilities
* **Model Caching:** In-memory caching for trained models
* **Load Balancing:** Ready for multiple application instances

## ****Security Implementation****

### ****Data Security****

* **Input Validation:** All user inputs sanitized and validated
* **API Rate Limiting:** Protection against abuse
* **Database Security:** MongoDB authentication and authorization
* **Error Handling:** Graceful failures without information disclosure

### ****Application Security****

* **Environment Variables:** Sensitive configuration externalized
* **CSRF Protection:** Flask-WTF integration for form security
* **XSS Prevention:** Template auto-escaping enabled
* **HTTPS Ready:** SSL/TLS support configuration

## ****Monitoring and Maintenance****

### ****Application Monitoring****

* **Performance Metrics:** Response time and throughput tracking
* **Error Logging:** Comprehensive error tracking and reporting
* **Model Performance:** Continuous accuracy monitoring
* **Database Health:** Storage and query performance monitoring

### ****Maintenance Procedures****

* **Model Retraining:** Automated retraining with new data
* **Data Backup:** Regular database backup procedures
* **Version Control:** Git-based code versioning and deployment
* **Documentation:** Complete technical and user documentation

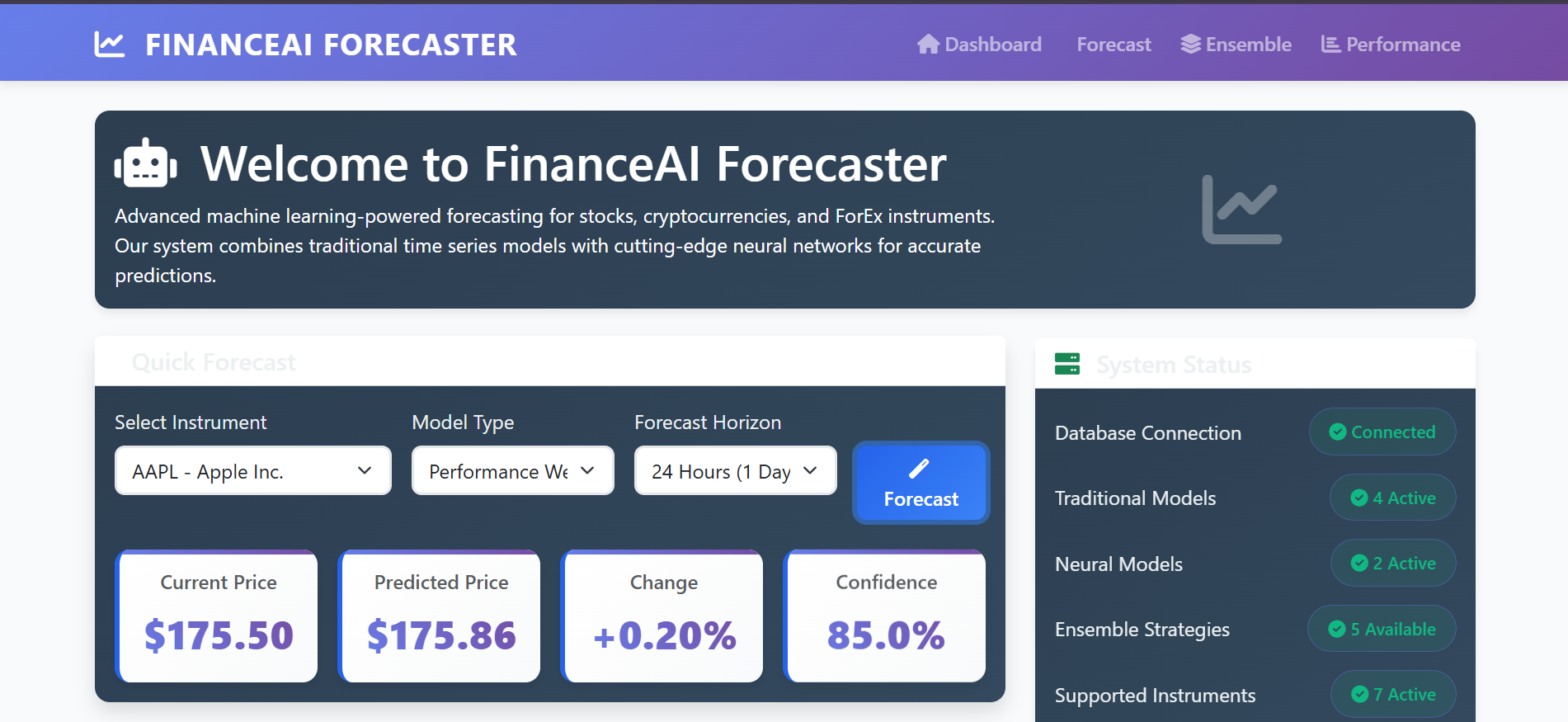
### ****Technical Skills Developed****

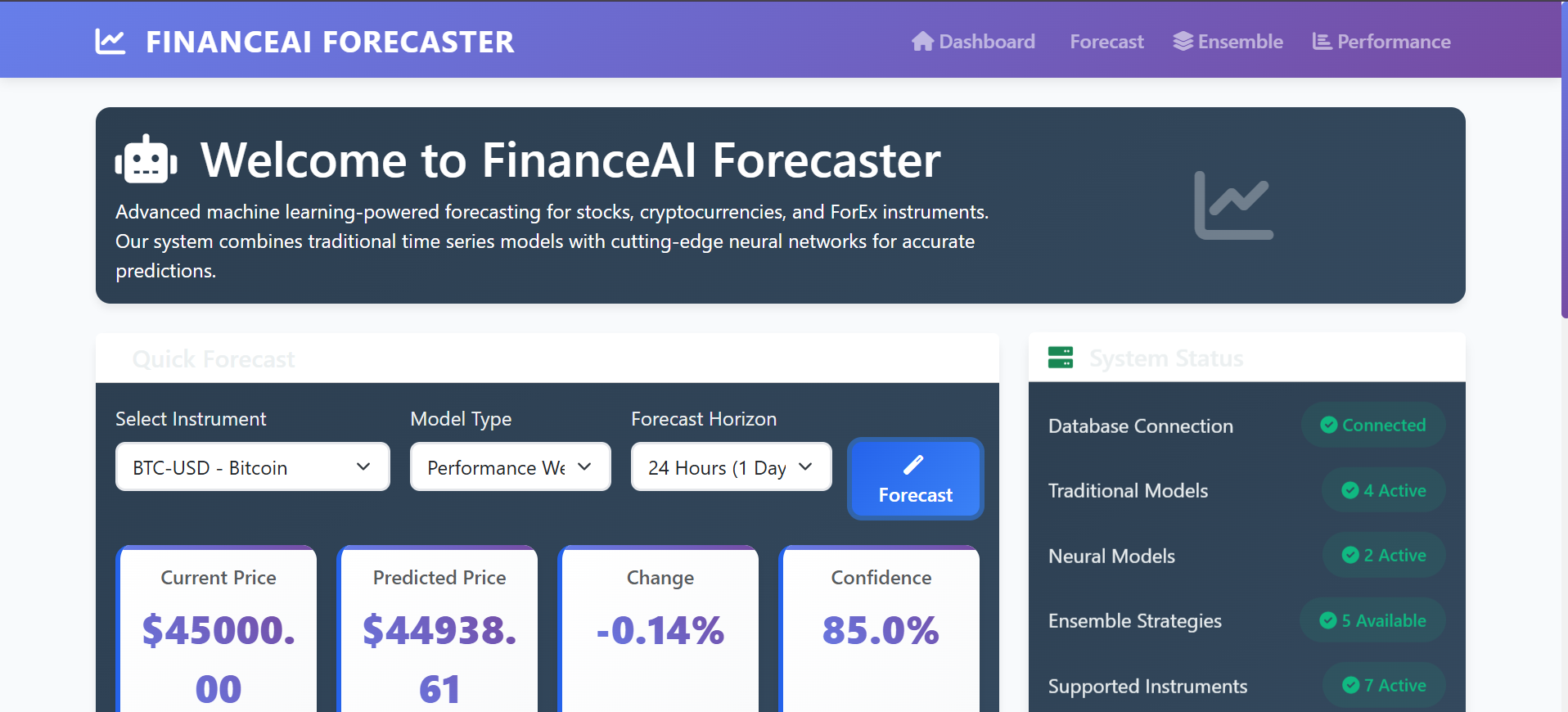
* **Full-stack Development:** End-to-end web application development
* **Machine Learning:** Implementation of diverse ML algorithms
* **Database Design:** Time series data modeling and optimization
* **Testing Methodologies:** Comprehensive test-driven development
* **Software Architecture:** Modular, maintainable code design

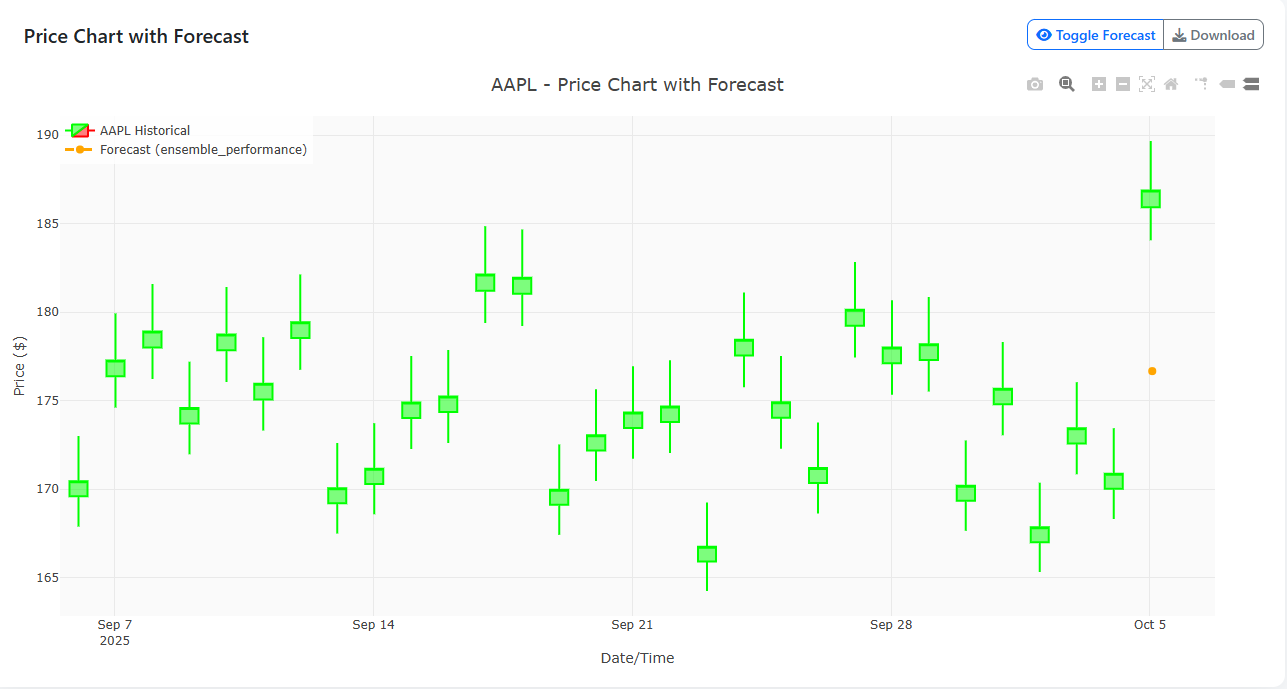
### ****Domain Knowledge Gained****

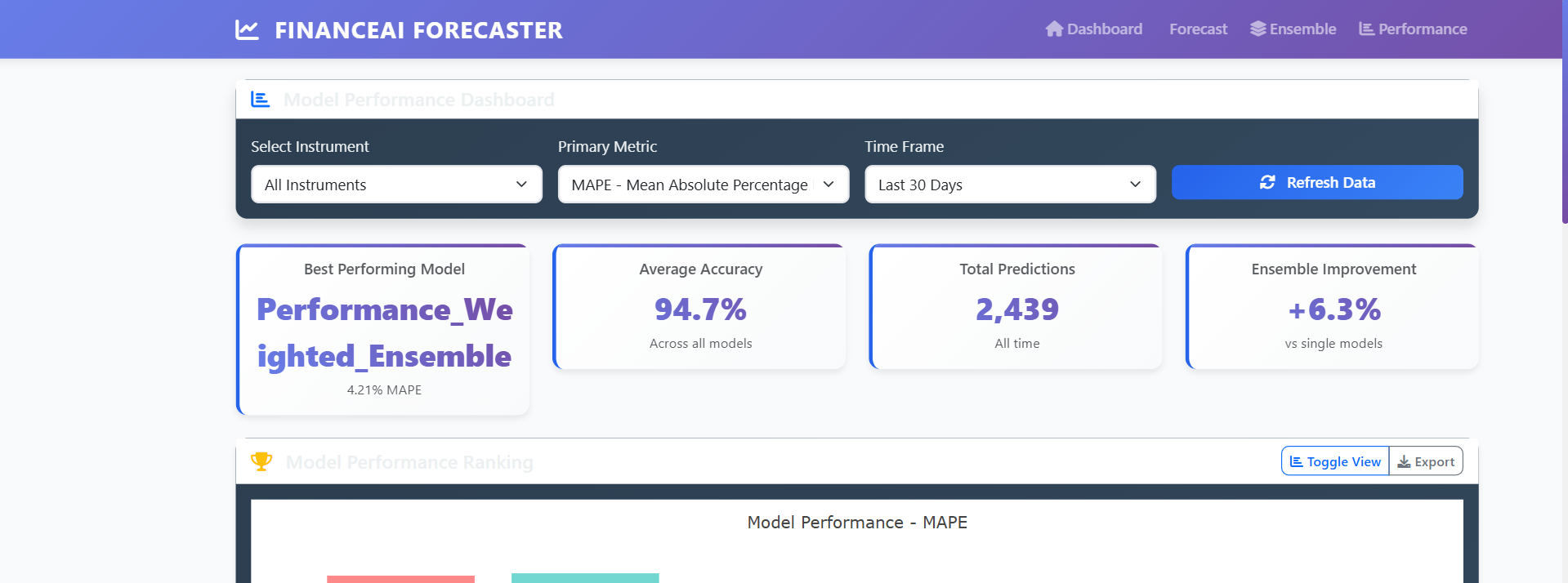
* **Financial Markets:** Understanding of price prediction challenges
* **Time Series Analysis:** Statistical and neural approaches to forecasting
* **Model Ensemble Methods:** Combining predictions for improved accuracy
* **Real-time Systems:** Handling live data streams and user interactions

Screenshots:

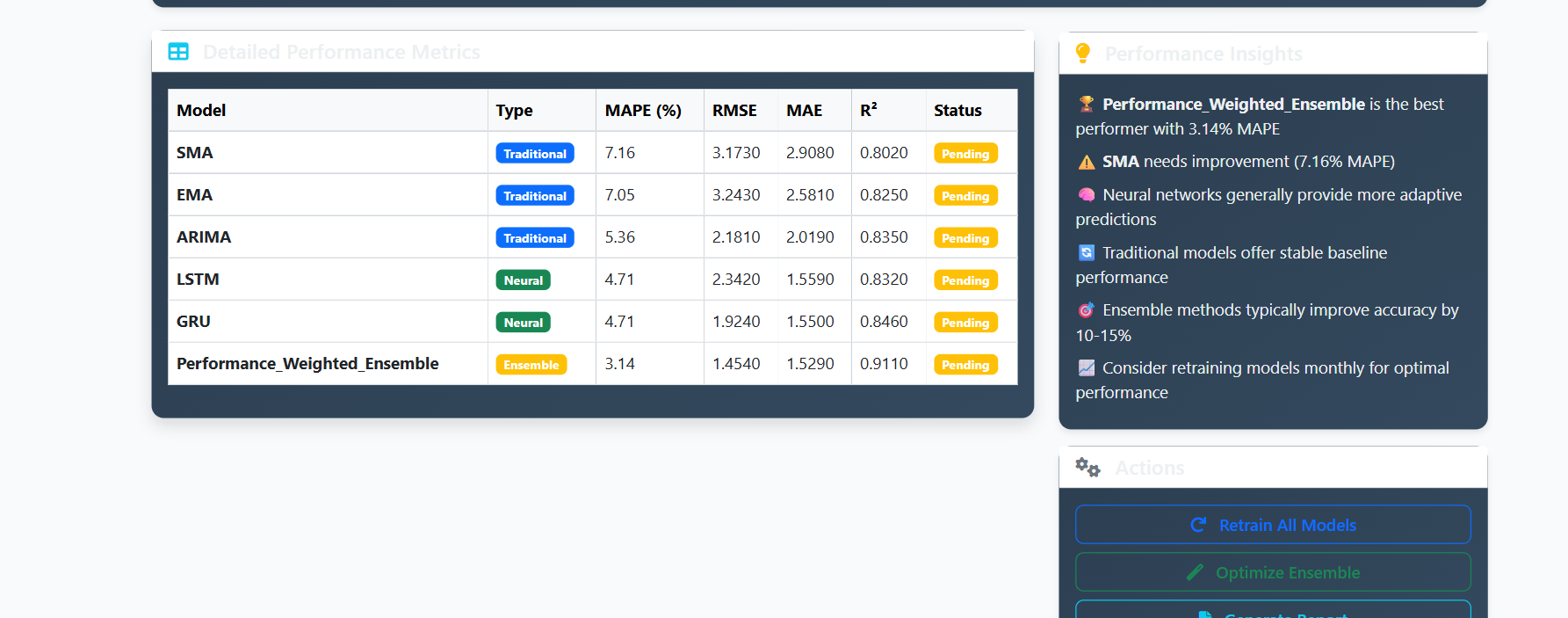


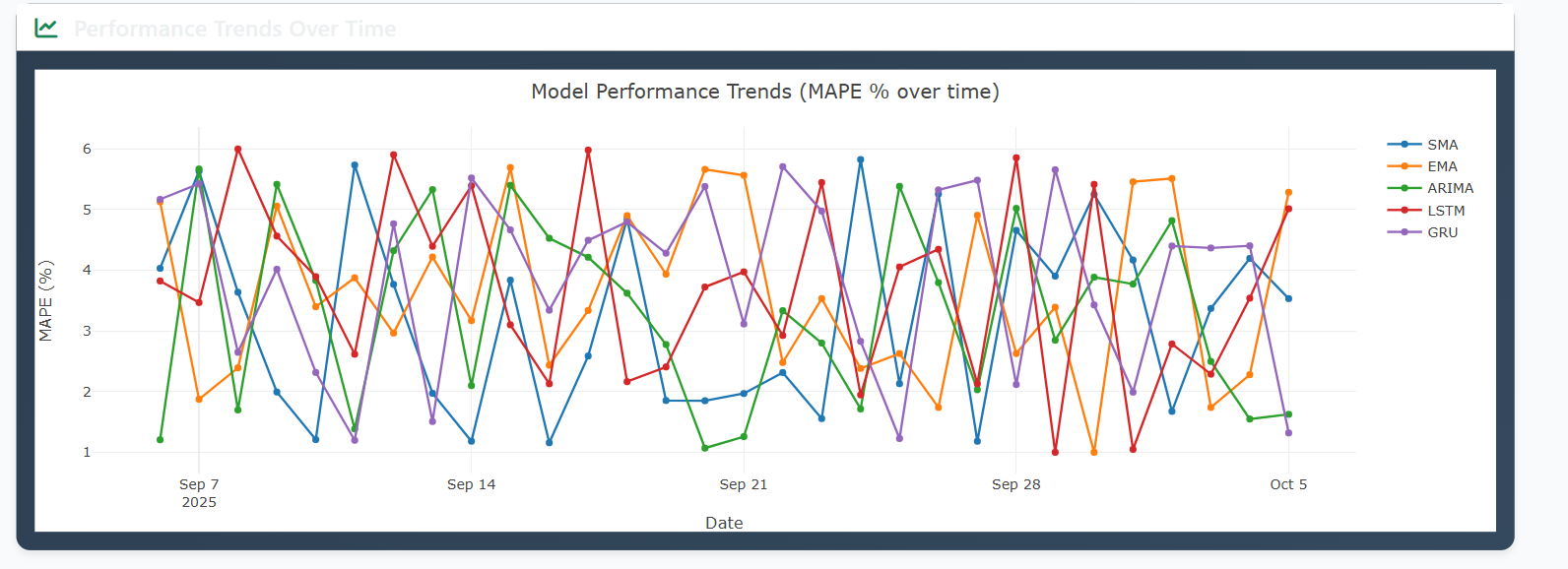


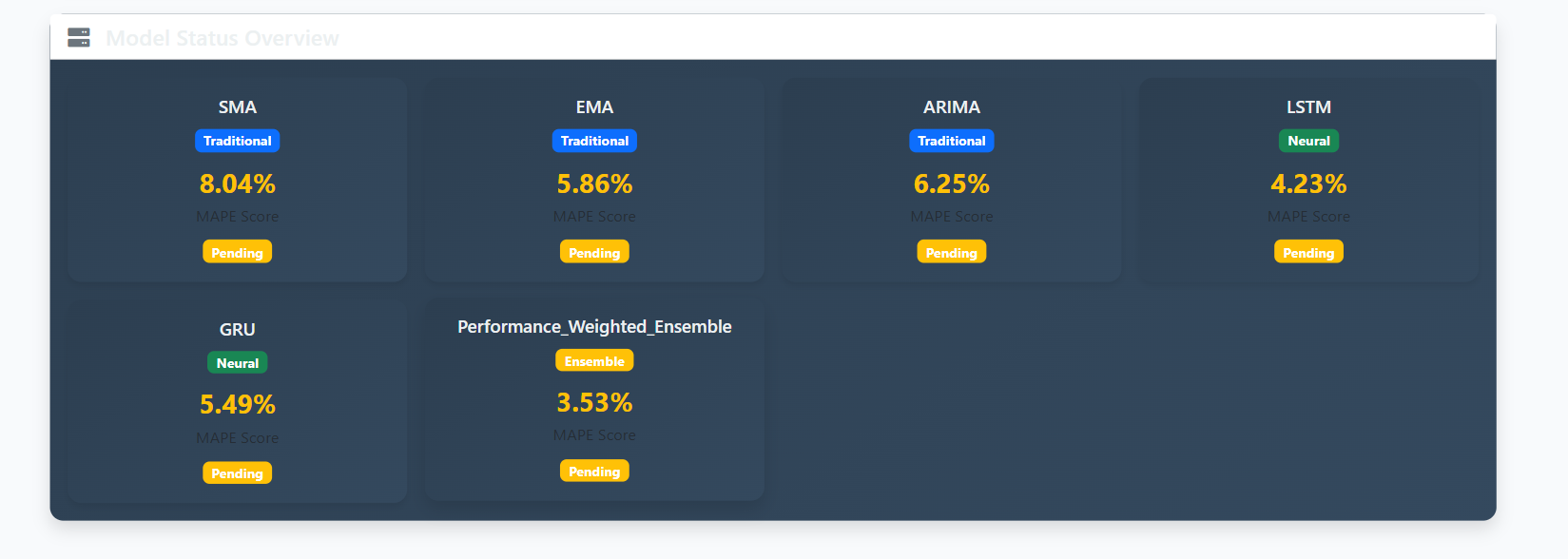












Architecture:

