**PROJECT Report**

On

**AI-POWERED AUTOMATED TAX COMPUTATION AND FILING SYSTEM**

Submitted in complete fulfilment of the requirements for the award of the degree of

**BACHELOR OF TECHNOLOGY**

**In**

**INFORMATION TECHNOLOGY**

**By**

**Gummalla Pavana Lakshmi Narasimha – 22261A1222**

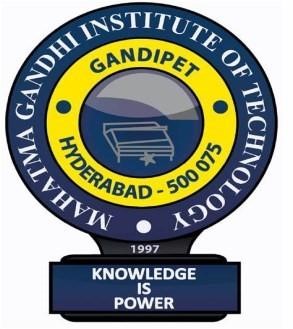
**Yetnesh Reddy C – 22261A1213**

Under the guidance of

**Mrs. U. Chaitanya**

**Assistant Professor**

Department of IT



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**MAHATMA GANDHI INSTITUTE OF TECHNOLOGY**

**(AUTONOMOUS)**

**(Affiliated to JNTUH, Hyderabad; Eight UG Programs Accredited by NBA; Accredited**

**by NAAC with ‘A++’ Grade)**

**Gandipet, Hyderabad, Telangana, Chaitanya Bharati (P.O), Ranga Reddy District, Hyderabad– 500075, Telangana**

**2024-2025**

**CERTIFICATE**

This is to certify that the **Minor Project** entitled **AI POWERED TAX COMPUTATION AND FILING SYSTEM** submitted by **Gummalla Pavana Lakshmi Narasimha(22261A1222), C. Yetnesh Reddy(22261A1213)** in complete fulfillment of the requirements for the Award of the Degree of Bachelor of Technology in Information Technology as specialization is a record of the bona fide work carried out under the supervision of **Mrs. U. Chaitanya** , and this has not been submitted to any other University or Institute for the award of any degree or diploma.

**Project Guide: Project Coordinator:**

**Mrs. U. Chaitanya Mrs. J Hima Bindu**

Assistant Professor Assistant Professor

Dept. of IT Dept. of IT

**EXTERNAL EXAMINAR Dr. D. Vijaya Lakshmi**

Professor and HOD

Dept. of IT

**DECLARATION**

We hear by declare that the **Mini Project** entitled **AI POWERED TAX COMPUTATION AND FILING SYSTEM** is an original and bona fide work carried out by us as a part of fulfilment of Bachelor of Technology in Information Technology, Mahatma Gandhi Institute of Technology, Hyderabad, under the guidance of **Mrs. U. Chaitanya,** **Assistant Professor**, Department of IT, MGIT.

No part of the project work is copied from books /journals/ internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the project work done entirely by us and not copied from any other source.

**Gummalla Pavana Lakshmi Narasimha – 22261A1222**

**C. Yetnesh Reddy - 22261A1213**

**ACKNOWLEDGEMENT**

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success. They have been a guiding light and source of inspiration towards the completion of the **Mini Project**.

We would like to express our sincere gratitude and indebtedness to our project guide **Mrs. U. Chaitanya**, **Assistant Professor**, Dept. of IT, who has supported us throughout our project with immense patience and expertise.

We are also thankful to our honourable Principal of MGIT **Prof. G. Chandramohan Reddy** and **Dr. D. Vijaya Lakshmi**, Professor and HOD, Department of IT, for providing excellent infrastructure and a conducive atmosphere for completing this **Project Stage-II** successfully.

We are also extremely thankful to our Project Coordinator **Mrs. J Hima Bindu**, Assistant Professor Department of IT, for their valuable suggestions and guidance throughout the course of this project.

We convey our heartfelt thanks to the lab staff for allowing us to use the required equipment whenever needed.

Finally, we would like to take this opportunity to thank our families for their support all through the work. We sincerely acknowledge and thank all those who gave directly or indirectly their support for completion of this work.

**Gummalla Pavana Lakshmi Narasimha – 22261A1222**

**C.Yetnesh Reddy -22261A1222**

**ABSTRACT**

Accurate cryptocurrency price prediction is a highly valuable asset in the trading world, offering significant advantages over strategies based on social media trends and speculative behavior. The ability to forecast price movements with high precision allows traders to make informed decisions, minimizing risks and optimizing returns. This study focuses on developing machine learning models trained on extensive historical data to predict the prices of leading cryptocurrencies like Bitcoin and Ethereum, which exhibit relatively higher stability and are often treated as stable investment options.

To ensure the reliability of the predictions, models are evaluated using key performance metrics such as R² (coefficient of determination), root mean squared error (RMSE), and mean absolute error (MAE). Three models—Linear Regression, LSTM Regression, and a proposed Voting Regression ensemble—were assessed across different datasets spanning one and two years for both Bitcoin and Ethereum.

Results demonstrate that the Voting Regression model consistently outperformed the others, achieving R² scores as high as 0.9898 for Bitcoin, with significantly lower RMSE and MAE values compared to traditional Linear Regression and standalone LSTM models. For Ethereum predictions, the Voting Regression model similarly excelled, reaching an R² of 0.9749, outperforming Linear Regression and closely matching or surpassing LSTM performance.

By providing more accurate forecasts, these models can significantly enhance trading strategies, helping investors navigate the highly volatile cryptocurrency market with greater confidence. Enhanced prediction accuracy reduces uncertainty, improves trade outcomes, and contributes to the development of more effective investment strategies and risk management practices in the crypto space.

**LIST OF FIGURES**

Fig. 3.2.1 Architecture of Bitcoin and Ethereum price prediction 12

Fig. 3.3.1 Level - 0 Data Flow Diagram 13

Fig. 3.3.2 Level - 1 Data Flow Diagram 14

Fig. 3.4.1.1 Use Case Diagram 20

Fig. 3.4.2.1 Class Diagram 22

Fig. 3.4.3.1 Activity Diagram For Predicting Price Of Coin 24

Fig. 3.4.4.1 Sequence Diagram 26

Fig. 3.4.5.1 Component Diagram 28

Fig. 6.1 Bitcoin Model Comparisons 54

Fig. 6.2 Ethereum Model Comparisons 55

Fig. 6.3 Initial page 55

Fig. 6.4 Registration Page 56

Fig. 6.6 Login page 56

Fig. 6.7 Main page 57

Fig. 6.8 Forecast Page 57

Fig. 6.9 Coin Select 58

Fig. 6.10 Forecast Period Select 58

Fig. 6.11 Bitcoin 6-month forecast 59

Fig. 6.12 Ethereum 6-month forecast 59

**LIST OF TABLES**

Table 2.1Literature Survey of Predicting prices of Bitcoin and Ethereum 9

Table 5.1Test Cases of Bitcoin and Ethereum Predictor 53

Table 6.1Bitcoin and Ethereum Performance Comparison54

**TABLE OF CONTENTS**

| **Chapter No** | **Title** | **Page No** |
| --- | --- | --- |
|  | **CERTIFICATE** |  |
|  | **DECLARATION** | **i** |
|  | **ACKNOWLEDGEMENT** | **ii** |
|  | **ABSTRACT** | **iii** |
|  | **LIST OF FIGURES** | **iv** |
|  | **LIST OF TABLES** | **v** |
| **1** | **INTRODUCTION** | **1** |
|  | 1.1 MOTIVATION | 1 |
|  | 1.2 PROBLEM STATEMENT | 1 |
|  | 1.3 EXISTING SYSTEM | 2 |
|  | 1.3.1 LIMITATIONS | 3 |
|  | 1.4 PROPOSED SYSTEM | 3 |
|  | 1.4.1 ADVANTAGES | 4 |
|  | 1.5 OBJECTIVES | 4 |
|  | 1.6 HARDWARE AND SOFTWARE REQUIREMENTS | 5 |
| **2** | **LITERATURE SURVEY** | **8** |
| **3** | **ANALYSIS AND DESIGN** | **11** |
|  | 3.1 MODULES | 11 |
|  | 3.2 ARCHITECTURE | 12 |
|  | 3.3 DATA FLOW DIAGRAM | 13 |
|  | 3.4 UML DIAGRAMS | 20 |
|  | 3.4.1 USE CASE DIAGRAM | 20 |
|  | 3.4.2 CLASS DIAGRAM | 22 |
|  | 3.4.3 ACTIVITY DIAGRAM FOR PREDICTING PRICE OF  COIN | 24 |
|  | 3.4.4 SEQUENCE DIAGRAM | 26 |
|  | 3.4.5 COMPONENT DIAGRAM | 28 |
|  | 3.5 METHODOLOGY | 15 |
| **4** | **CODE AND IMPLEMENTATION** | **30** |
|  | 4.1 CODE | 30 |
|  | 4.2 IMPLEMENTATION | 49 |
| **5** | **TESTING** | **52** |
| **6** | **RESULTS** | **54** |
| **7** | **CONCLUSION AND FUTURE ENHANCEMENTS** | **60** |
|  | 7.1 CONCLUSION | 60 |
|  | 7.2 FUTURE ENHANCEMENTS | 60 |
|  | **REFERENCES** | **61** |

1. **INTRODUCTION**

**1.1 MOTIVATION**

Cryptocurrencies, particularly Bitcoin and Ethereum, have emerged as revolutionary digital assets that have transformed the global financial landscape. Their decentralized architecture, enabled by blockchain technology, ensures transparency, immutability, and security, making them an attractive alternative to traditional financial instruments. Over the years, cryptocurrencies have garnered immense popularity among a wide range of users, including individual investors, institutional traders, and technology enthusiasts. However, one of the defining characteristics of cryptocurrencies - their extreme price volatility - also represents their greatest challenge. This volatility introduces significant risks for investors and limits the usability of cryptocurrencies as a stable store of value.

In the current financial ecosystem, accurate price forecasting has become a crucial factor for mitigating these risks and ensuring better financial outcomes for stakeholders. Existing forecasting models often prioritize short-term predictions and are constrained by limited datasets, hindering their ability to deliver accurate and reliable results over longer periods. Additionally, the fast-paced and highly reactive nature of cryptocurrency markets makes it imperative for models to dynamically adapt to sudden changes in market conditions. The inability of existing systems to address these challenges effectively has created a demand for solutions that combine real-time data integration with advanced prediction techniques.

This project is motivated by the need to overcome these limitations by building a more robust and adaptable forecasting framework.

**1.2 PROBLEM STATEMENT**

The rapid growth of the cryptocurrency market has brought new opportunities for investors and financial institutions. However, the extreme volatility of cryptocurrencies like Bitcoin and Ethereum makes it exceedingly difficult to predict their future price movements with accuracy. Traditional methods of financial forecasting often fail to address the unique challenges posed by cryptocurrencies, including long term predictions, reliance on speculative trends, and highly dynamic trading volumes.

The problem is further compounded by the lack of comprehensive systems that can integrate live market data and adapt to sudden, significant changes in market behaviour.

Most existing cryptocurrency price prediction models focus exclusively on short-term forecasting, often limited to a prediction horizon of a few days or weeks. These models, while useful for day traders and short-term investors, fail to meet the needs of users seeking reliable long-term forecasts. Moreover, many models are built on outdated or static datasets that fail to capture the ever-evolving nature of the cryptocurrency market. As a result, the predictions provided by these models often lack the accuracy and adaptability needed for effective decision-making in a volatile market environment.

This project seeks to address these gaps by developing a robust, adaptable, and accurate cryptocurrency price prediction system. By combining real-time data acquisition with advanced machine learning approaches, such as LSTM for sequential data, logistic regression for trend classification, and voting regressors for ensemble modelling, the proposed system aims to significantly enhance prediction accuracy. Additionally, the system will provide long-term forecasting capabilities, offering actionable insights for a broader range of investors. Ultimately, this project aims to create a reliable tool that bridges the gap between cryptocurrency price unpredictability and the need for informed investment strategies.

**1.3 EXISTING SYSTEM**

Existing cryptocurrency price prediction systems focus heavily on short-term and mid-term forecasting. They primarily target stablecoins such as Bitcoin, Ethereum, Binance Coin, and Litecoin due to their high liquidity and significant market activity. These systems typically predict price movements for horizons of 1 day, 7 days, 14 days, 30 days, and, in some cases, up to 60 days. Such intervals cater to the needs of traders and investors seeking actionable insights within a relatively short timeframe.

The models used for these predictions are predominantly based on Long Short-Term Memory (LSTM) networks and their variations. LSTMs are well-suited for time-series data, allowing them to capture temporal dependencies and trends effectively.

These systems use training datasets spanning 2–3 years of historical data, which include parameters like opening price, closing price, high and low prices, trading volume, and market sentiment. Bitcoin, being the most researched cryptocurrency, often yields higher prediction accuracy due to its relatively stable market behaviour and the abundance of historical data available for training.

The predictive accuracy of these systems typically ranges between 50–70%, with Bitcoin generally achieving better performance compared to other cryptocurrencies.

**1.3.1 Limitations**

* **Limited Adaptability to Rapid Market Changes**: While the model attempts to preserve trends and correct biases, the rapid volatility in cryptocurrency markets may still challenge the model's ability to adapt quickly to sudden changes, potentially leading to less accurate predictions during high volatility periods.
* **Limited Predictive Horizons:** The system primarily focuses on short-term and mid-term predictions (e.g., 1, 7, 14, 30, and 60 days). This is not be suitable for users interested in long-term forecasting.

**1.4 PROPOSED SYSTEM**

The proposed cryptocurrency price prediction system focuses on enhancing accuracy and adaptability by addressing existing limitations in traditional models. A key feature of the system is the ability to update its dataset daily as new data becomes available. This ensures that the model remains aligned with the latest market trends and fluctuations. By leveraging reliable sources such as Yahoo Finance, the system retrieves essential metrics like opening and closing prices, high and low values, and trading volumes. These updates allow the model to incorporate fresh information seamlessly, maintaining its relevance and responsiveness to real-time market conditions.

To improve the accuracy of long-term predictions, the system adopts a mixed-model approach that combines outputs from various regression models. For instance, it uses techniques such as a Voting Regressor, which aggregates predictions from different algorithms like Random Forest, Gradient Boosting, and Support Vector Regression (SVR). This hybrid approach ensures a more balanced and robust forecast by integrating the strengths of multiple models. Such a framework is particularly effective for longer forecasting periods, where a single algorithm might struggle to capture complex market dynamics.

Additionally, the system takes advantage of larger datasets to improve the overall performance of its predictions. By training on extended periods of historical data, it gains a deeper understanding of long-term trends and patterns. This not only boosts accuracy but also equips the system to handle the inherent volatility and noise present in cryptocurrency markets. Through a combination of daily updates, mixed-model forecasting, and the use of comprehensive datasets, the proposed system aims to provide reliable predictions for both short- and long-term horizons, catering to the needs of traders and investors in a rapidly changing market environment.

**1.4.1 ADVANTAGES**

* **Real-Time Adaptability**: The system updates in real time based on new data, ensuring it remains aligned with the latest market trends and developments, which enhances performance in volatile market conditions where price movements change rapidly.
* **Long-Term Forecasting Capability**: It predicts for extended time periods using mixed-model frameworks like Voting Regressors, combining multiple regression models to account for both short-term fluctuations and long-term trends, enabling traders and investors to make better-informed strategic decisions.
* **Increased Reliability**: The constantly updating dataset ensures predictions are based on the most recent and comprehensive information, reducing the risk of relying on outdated or incomplete data and improving overall accuracy and robustness.

**1.5 OBJECTIVES**

* **Prediction Based on Live Data**: Enable the system to generate predictions using live data, ensuring accuracy and relevance in rapidly changing market conditions.
* **Integration of Real-Time Data Sources**: Utilize real-time data from reliable sources like Yahoo Finance to keep the dataset updated and adaptable to market trends.
* **Focus on Long-Term Predictions**: Develop robust forecasting capabilities for long-term horizons, allowing users to make strategic and informed investment decisions.

**1.6 HARDWARE AND SOFTWARE REQUIREMENTS**

**Software Requirements**

** Software:**

The system uses Anaconda, a powerful distribution for Python and data science applications. Anaconda provides an integrated environment to manage libraries, dependencies, and packages, making it ideal for developing and running machine learning models.

 **Primary Language**:

The system is primarily developed in Python, a versatile programming language well-suited for machine learning, data analysis, and backend development. Python offers extensive libraries such as NumPy, pandas, TensorFlow, and Scikit-learn, which are essential for model building and data processing.

 **Frontend Framework**:

The user interface is built using Flask, a lightweight web framework for Python. Flask allows seamless integration of the machine learning backend with the frontend, enabling real-time interaction between users and the prediction models.

 **Backend Framework**:

The core machine learning models and data processing tasks are implemented in Jupyter Notebook, an open-source web-based environment that supports interactive computing. Jupyter Notebook provides an intuitive interface for coding, debugging, and visualizing results.

 **Database**:

The system uses SQLite3 as the database for storing and managing data. SQLite3 is a lightweight, serverless database management system that is highly efficient for small to medium-scale applications. It is used to store historical data, user preferences, and prediction outputs.

 **Frontend Technologies**:

The frontend of the system incorporates the following technologies to ensure a responsive and visually appealing user interface:

* **HTML**: For structuring the web pages and content.
* **CSS**: For styling and enhancing the visual presentation of the interface.
* **JavaScript**: For adding interactivity and dynamic features to the user interface.
* **Bootstrap 4**: A responsive design framework to ensure the application works seamlessly across various devices and screen sizes.

**Hardware Requirements**

 **Operating System**:

The system is designed to run on **Windows** operating systems. This ensures compatibility with the development tools and frameworks used in the project.

 **Processor**:

A processor with a performance level of **Intel i5 or above** is recommended. This ensures the system can handle the computational load required for data processing, model training, and prediction generation.

 **RAM**:

The system requires a minimum of **8GB RAM or higher** to manage multiple processes efficiently, including real-time data integration, feature engineering, and model execution. Higher RAM capacity is recommended for faster performance and smooth multitasking.

** Hard Disk**:

A minimum of **25GB of storage space** is required for the installation of software like Anaconda and libraries, storage of datasets, and saving model outputs. Additional storage may be needed depending on the size of the dataset and user-generated data.

1. **LITERATURE SURVEY**

Rafi et al. explored a feature selection and weighting approach integrated with a Bi-Directional Long Short-Term Memory (Bi-LSTM) model to enhance the accuracy of cryptocurrency price forecasting. The Bi-LSTM model captures both past and future price movements, crucial for time-series data like cryptocurrency prices. The study demonstrated that this approach could improve prediction performance by correcting model biases, but its computational demands and the need for extensive datasets posed challenges, making it difficult to implement in real-time applications. Despite the improvements in prediction accuracy, the high computational costs associated with tuning the Bi-LSTM model hinder its scalability for broader use [1].

Shamshad et al. focused on forecasting and trading stable cryptocurrencies like Litecoin and Binance Coin using machine learning and deep learning models. Their study showed success in predicting prices for very stable cryptocurrencies like Tether (USDT), but their approach was less effective for volatile assets like Bitcoin and Ethereum. They highlighted the importance of tailoring models to specific types of cryptocurrencies, as volatile currencies require more adaptive systems that can handle rapid price changes. Furthermore, the models in their study were not able to update quickly enough with real-time data, limiting their usefulness in fast-paced market conditions [2].

Kim et al. incorporated on-chain data, such as transaction volume and mining difficulty, into their deep learning-based model for cryptocurrency price prediction. This approach demonstrated that integrating blockchain-based metrics could lead to more accurate predictions, particularly when forecasting long-term trends. However, the reliance on on-chain data restricts the model’s applicability to cryptocurrencies with sufficient blockchain data, such as Bitcoin and Ethereum. This limitation reduces the generalizability of the model to other digital assets that may lack extensive on-chain data, making it less versatile for broader market analysis [3].

Shahbazi and Byun explored reinforcement learning as a method to improve cryptocurrency price prediction. They found that reinforcement learning could help models adapt to market changes by continuously learning from real-time data. However, the model's reliance on large datasets and high computational resources posed challenges in terms of both training time and cost. The reinforcement learning model demonstrated flexibility and responsiveness to market changes but required significant computational power, making it difficult for practical, real-time applications in cryptocurrency forecasting [4].

Tanwar et al. proposed a deep learning-based approach that incorporated the inter-dependent relationships between different cryptocurrencies. Their model accounted for the correlations between Bitcoin, Ethereum, and other cryptocurrencies, which is important because the price of one digital asset often influences the others. The approach showed promise, but its success heavily depended on the quality and quantity of the input data. Furthermore, the ability of the model to generalize across different cryptocurrencies with varying levels of volatility remains an area for further research [5].

Jay et al. introduced a stochastic neural network model to handle the uncertainty and noise in cryptocurrency price data. This model aimed to simulate multiple possible outcomes and provide more robust predictions, which is essential given the high volatility in the cryptocurrency market. Although the model performed well over time, it struggled to predict sudden price spikes or crashes, which are common in cryptocurrency markets. The study suggested that combining stochastic models with other machine learning techniques, such as ensemble methods, could improve both the accuracy and responsiveness of price predictions [6]

Table 2.1 Literature Survey of Predicting prices of Bitcoin and Ethereum

|  |  |  |
| --- | --- | --- |
| **Reference** | **Method** | **Limitations** |
| [1] | Rafi et al. aim to enhance the accuracy of cryptocurrency price forecasting by using a feature selection and weighting approach with Bi-Directional Long Short-Term Memory (Bi-LSTM). The model is tested on both Bitcoin and Ethereum using historical data. | Limited Predictive horizons:  It focuses on short term and midterm predictions  Model degradation over time: As market conditions change, the model's bias correction technique might not remain effective, leading to outdated predictions. |
| [2] | Shamshad et al. explore the prediction of stable cryptocurrency prices using a variety of machine learning (ML) and deep learning (DL) models, including ARIMA, Support Vector Regressor (SVR), and LSTMs. | Short prediction horizon:  The models focus on short-term predictions (10 days), which might not be suitable for long-term forecasting​ |
| [3] | Kim et al. present a deep learning model focused on predicting cryptocurrency prices using on-chain data. The model utilizes advanced machine learning techniques to forecast future prices based on historical blockchain data. | Real-time adaptability:  The model might not perform well in real-time predictions due to the time required for processing large amounts of on-chain data, making it less effective for rapid market responses |

|  |  |  |
| --- | --- | --- |
| **Reference** | **Proposed Method** | **Limitations** |
| [4] | Shahbazi and Byun propose a machine learning model utilizing reinforcement learning (RL) to predict the prices of Monero and Litecoin. The RL model is trained on cryptocurrency market data and predicts short- and long-term price movements. | Computational cost:  Using Reinforcement learning algorithms can be resource-intensive, requiring significant computational power and time for training.  Overfitting risk:  The RL model might overfit on specific historical data patterns, performing poorly when market conditions change. |
| [5] | Tanwar et al. propose a deep learning scheme that incorporates inter-dependent relationships among cryptocurrencies to improve prediction accuracy. This analyzes the interrelations between different cryptocurrencies like Bitcoin, Ethereum, and others to generate a more comprehensive prediction model. | Complexity:  The interdependence model adds complexity to the training process, potentially requiring more computational resources.  Limited Predictive horizons:  It focuses only on short term predictions |
| [6] | Jay et al. present a stochastic neural network model based on random walk theory to predict cryptocurrency prices. The model adds stochastic layers to standard neural networks like MLP and LSTM to capture the volatility of the cryptocurrency market. | The stochastic nature of the model may introduce unpredictability, and it might not perform well under stable market conditions. |

1. **ANALYSIS AND DESIGN**

The cryptocurrency market is highly volatile and unpredictable, presenting significant challenges for traders and investors looking to make informed decisions. This project aims to develop an advanced cryptocurrency price prediction system designed to address these challenges by leveraging real-time data integration, machine learning techniques, and robust forecasting models. The system is built to provide both short-term and long-term price predictions, enabling users to make smarter trading and investment decisions based on accurate and up-to-date insights.

The core of the project lies in its ability to integrate live data sources, such as Yahoo Finance, to create a continuously updated and reliable dataset. By pulling the latest information on cryptocurrency prices, trading volumes, and other key metrics, the system ensures that its predictions remain relevant in rapidly changing market conditions. This real-time adaptability is essential in the crypto space, where prices can fluctuate dramatically within minutes due to news events, regulatory announcements, or changes in market sentiment.

To deliver accurate predictions, the system employs a mixed-model approach. This involves use of a mixed model, such as Voting Regressors, which aggregate the outputs of multiple machine learning algorithms like Logistic Regression and Decision trees.

**3.1 MODULES**

**User Interaction Module**

* **Functions:** Allows the user to select the cryptocurrency (Bitcoin or Ethereum) and view the prediction results. Displays predicted prices in user-friendly line charts and displays predicted prices in incremental format.
* **Input:** Selected coin (Bitcoin/Ethereum), Predicted price data.
* **Output:** Displays predicted prices, line charts, and incremental milestones (next day, next week prices, etc).

**Preprocessing Module**

* **Functions:**  Collects real-time data from Yahoo Finance and retrieves historical data for model training. Generates time frames and calculates moving averages for trend identification.
* **Input:** Coin selected by the user, Raw data (real-time and historical).
* **Output:** Processed data with features like moving averages for multiple time frames.

**Prediction Module**

* **Functions:** Predicts future prices using the Voting Regressor model based on the processed real-time data.
* **Input:** Processed real-time data.
* **Output:** Predicted price data for multiple milestones and line chart for 6 months.

**Visualization Module**

* **Functions:** Displays predicted prices in user-friendly line charts and displays predicted prices in incremental format.
* **Input:** Predicted price data.
* **Output:** Line charts and incremental prices.

**3.2 ARCHITECTURE**

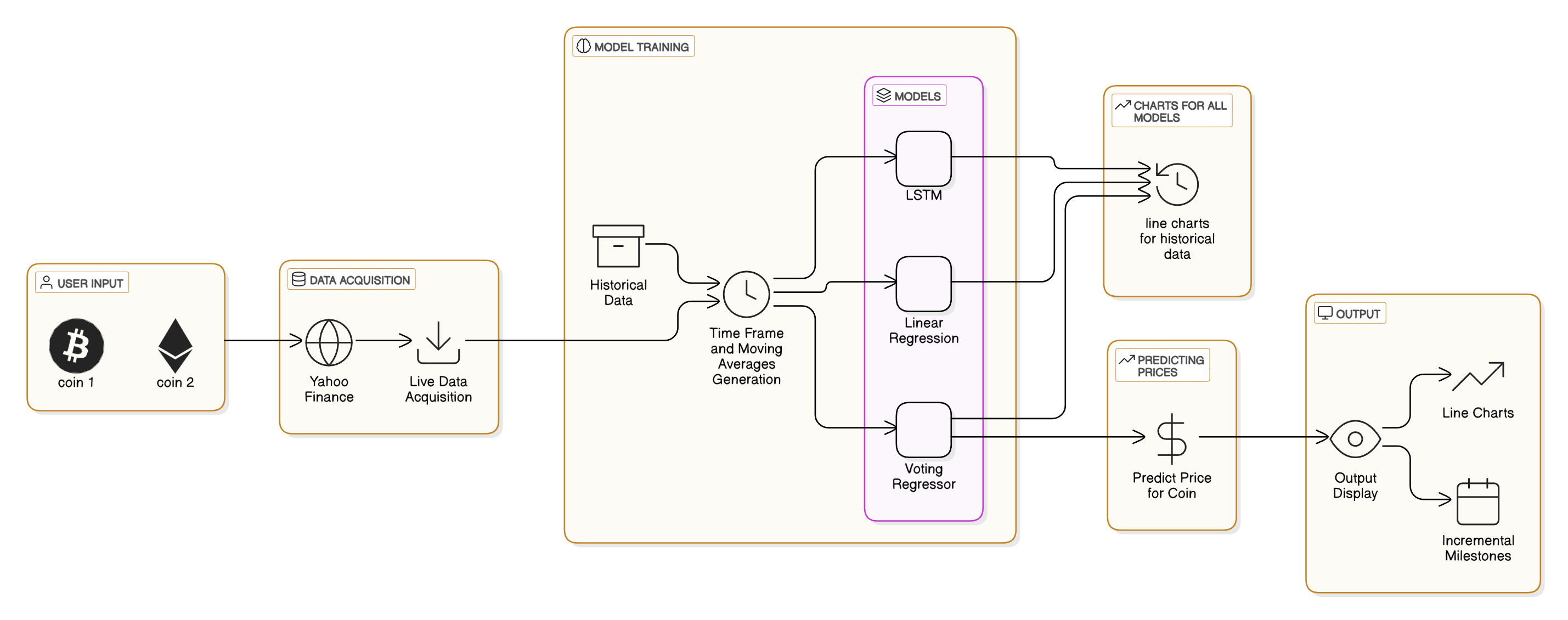
****

Fig. 3.2.1 Architecture of Bitcoin and Ethereum price prediction

The architecture of the system as shown in Fig. 3.2.1, is designed for predicting the prices of Bitcoin and Ethereum begins with the **Data Acquisition** component. This involves fetching historical price data for both cryptocurrencies from Yahoo Finance. Once the data is acquired, it is sent to the main model for training.

In the **Main Model** phase, the system calculates the moving averages for the acquired data to identify trends. The model is then trained using this historical data and subsequently used to predict the prices for the next six months. The trained model's results are then displayed to the user. Live training occurs using the Voting Regressor model, continually updating and refining its predictions based on the latest available data. The trained model's results are then displayed to the user

The **Main Website** component plays a crucial role in presenting the results. It displays the accuracy of different models using line charts, shows the predicted prices for the next six months as a line chart, and highlights the incremental steps of the predicted prices for the selected cryptocurrency.

Users interact with the system through the **User Input** component, where they can select either Bitcoin or Ethereum to view the price forecast.

The system incorporates various **Trained Models**, including Long Short-Term Memory (LSTM) results, Logistic Regression results, and Voting Regressor results, to visually show the reason for using Voting Regressor.

**3.3 DATA FLOW DIAGRAMS**

**Level - 0**

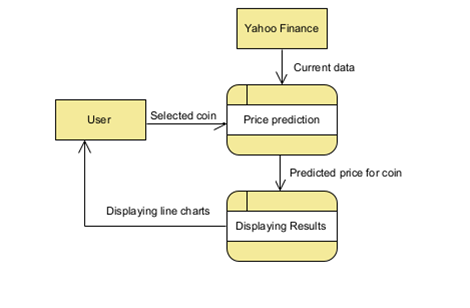


Fig. 3.3.1 Level - 0 Data Flow Diagram

This Level-0 Data Flow Diagram (Fig. 3.3.1) represents a system designed to predict the prices of Bitcoin and Ethereum. The process begins when the user selects either Bitcoin or Ethereum as the cryptocurrency they wish to analyse. Once the selection is made, the system retrieves the current market data for the chosen cryptocurrency from Yahoo Finance. This data is then fed into the Price Prediction process, which forecasts the future price of the selected cryptocurrency. Finally, the system displays the predicted price to the user along with line charts, providing a visual representation of the predicted price trends over time.

**Level – 1**

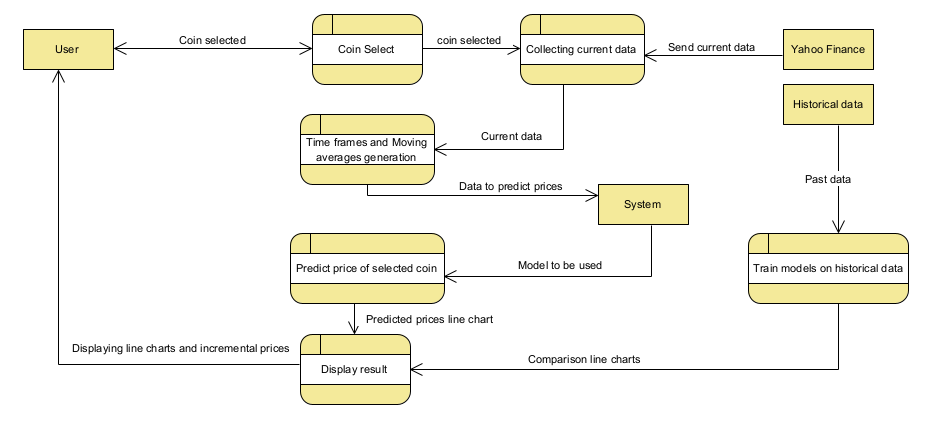
****

Fig. 3.3.2 Level - 1 Data Flow Diagram

This Level-1 Data Flow Diagram (Fig. 3.3.2) represents a system for predicting the prices of cryptocurrencies such as Bitcoin and Ethereum. The process begins with the user selecting a coin of interest.

The selected coin is passed to a system that collects current data from external sources like Yahoo Finance and incorporates historical data for training predictive models. The collected data is used to generate time frames and moving averages, which serve as inputs for prediction.

The predictive models, trained on historical data, are employed to forecast the price of the selected cryptocurrency. The predicted prices are visualized as line charts and compared against historical data for accuracy. Finally, the results, including line charts and incremental price changes, are displayed to the user. The system ensures a seamless flow of data and integrates real-time analytics with historical insights to provide accurate predictions.

**3.4 UML DIAGRAMS**

**3.4.1 USE CASE DIAGRAMS**

A use case diagram is a visual representation that depicts the interactions between various actors and a system, capturing the ways in which users or external entities interact with the system to achieve specific goals. It is an essential tool in system analysis and design, often used in software engineering and business analysis. In a use case diagram, actors are entities external to the system that interact with it, and use cases are specific functionalities or features provided by the system as seen in Fig. 3.4.1.1. These interactions are represented by lines connecting actors to use cases. The diagram helps to illustrate the scope and functionality of a system, providing a high-level view of how users or external entities will interact with it.

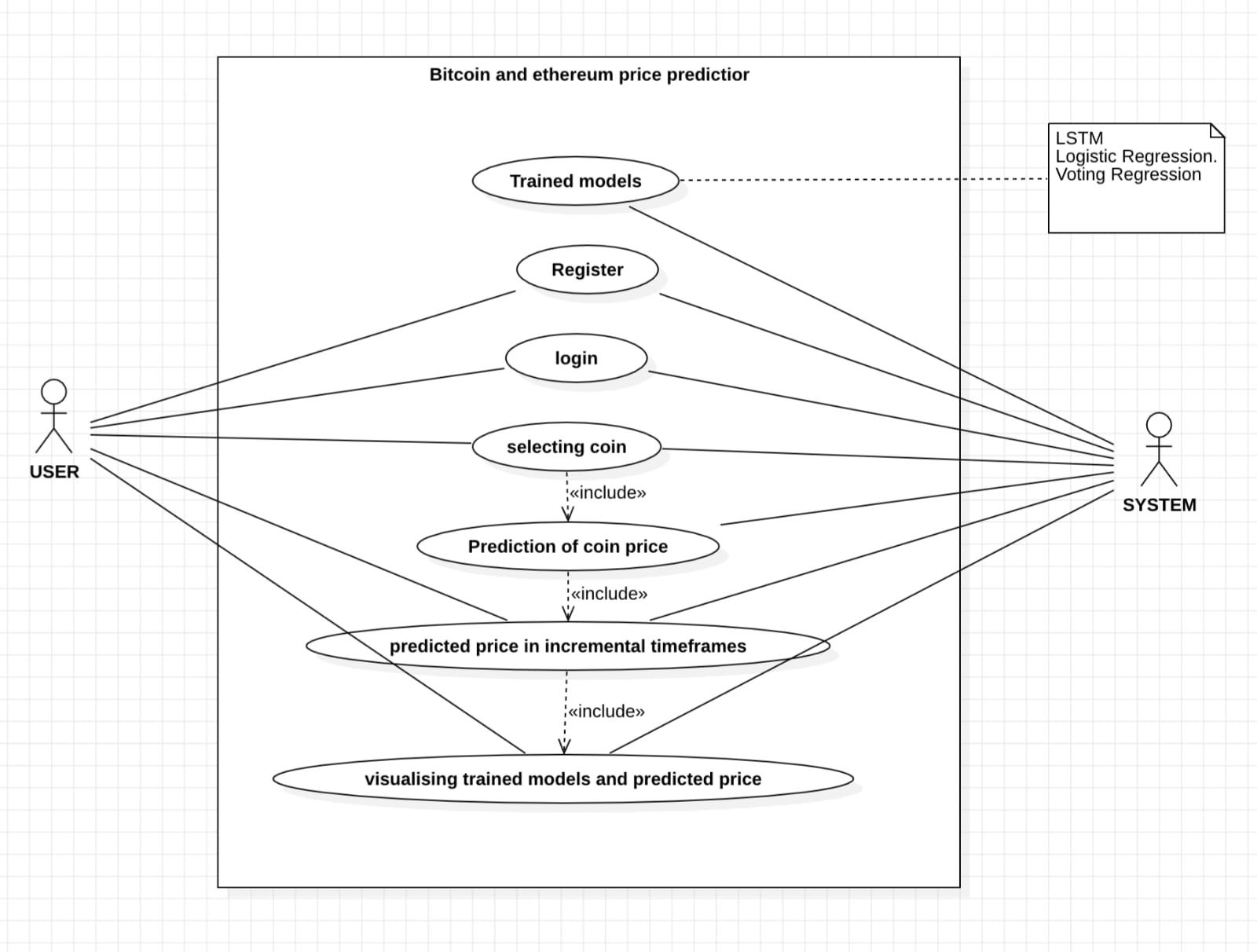


Fig. 3.4.1.1 Use Case Diagram

**Actors**

* **User**  
  Represents the person interacting with the system.
* **System**  
  Represents the back-end system performing operations and computations.

**Use Cases**

* **Register**

The user can register on the platform to gain access to features.

* **Login**

After registering, the user logs into the system to access functionalities.

* **Selecting Coin**

The user chooses a cryptocurrency (Bitcoin or Ethereum) for price prediction.

* **Prediction of Coin Price**

The system predicts the price of the selected cryptocurrency based on trained models.

* **Predicted Price in Incremental Timeframes**

Provides price predictions for multiple timeframes (e.g., hourly, daily, weekly).

* **Visualizing Trained Models and Predicted Price**

Displays the trained machine learning models and their predictions through graphs or charts.

**3.4.2 CLASS DIAGRAM**

A class diagram is a visual representation that models the static structure of a system, showcasing the system's classes, their attributes, methods (operations), and the relationships between them as seen in Fig. 3.4.2.1. It is a key tool in object-oriented design and is commonly used in software engineering to define the blueprint of a system.

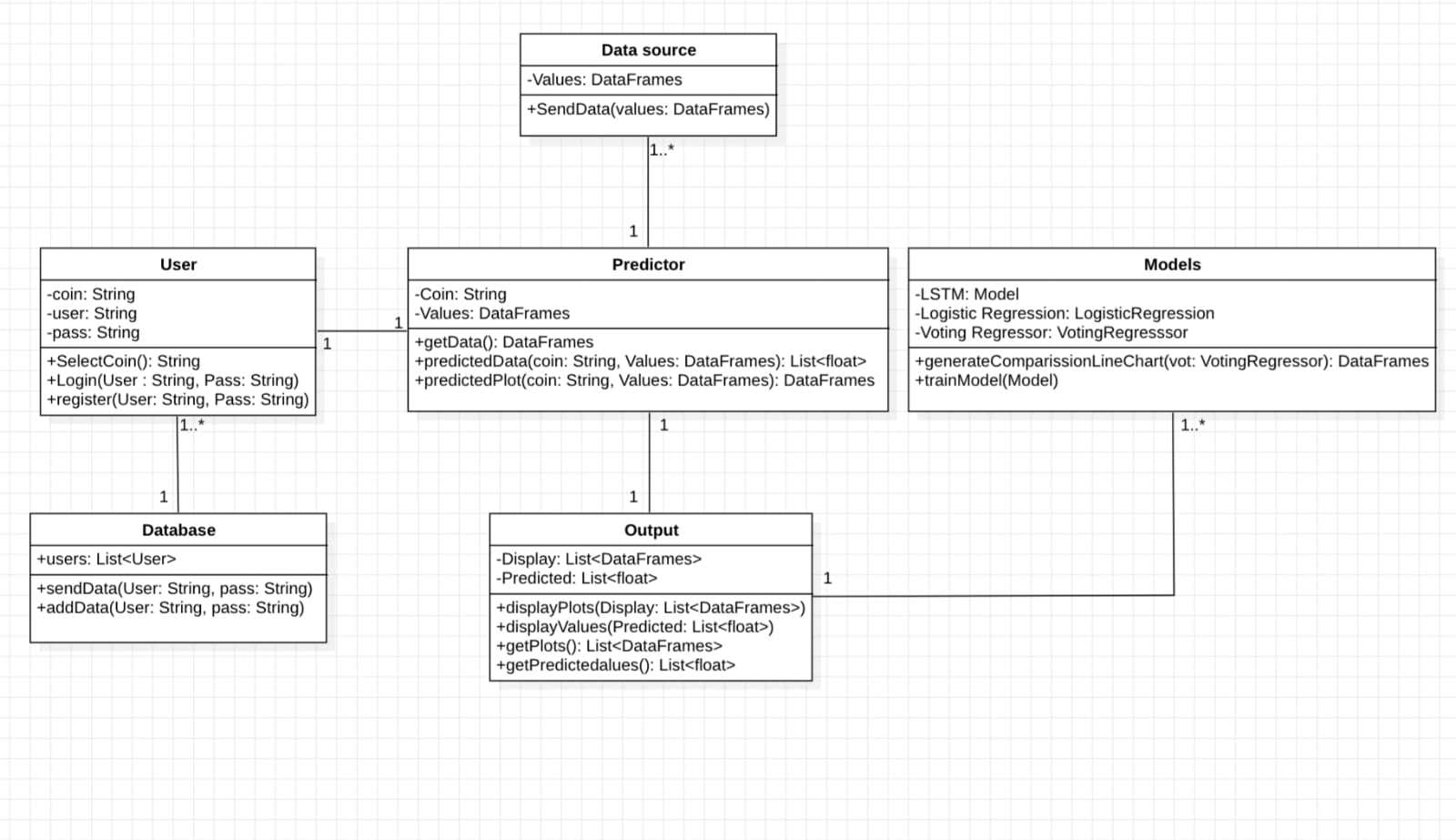


Fig. 3.4.2.1 Class Diagram

**Relationships**

* **User → Database**
  + The user interacts with the database for login and registration. A user can log in or register, and the database handles these operations.
* **Database → Predictor**
  + The predictor retrieves data from the database to perform predictions.
* **Predictor → Data Source**
  + The predictor uses the data source to fetch historical data for model training and predictions.
* **Predictor → Models**
  + The predictor utilizes different machine learning models (e.g., LSTM, Logistic Regression, Voting Regressor) to perform predictions.
* **Predictor → Output**
  + The predictor sends predicted data and plots to the output class for visualization.
* **Models → Output**
  + Models directly contribute to generating comparison charts and predictive visualizations.

**System Flow**

* **User Interaction**:
  + A user logs in or registers via the User class.
  + The Database class verifies the credentials or stores new users.
* **Data Fetching**:
  + The Predictor fetches data from the Data Source.
* **Prediction**:
  + The Predictor uses trained Models to predict cryptocurrency prices.
  + The results are passed to the Output class.
* **Visualization**:
  + The Output class generates plots and displays predicted values.

**3.4.3 ACTIVITY DIAGRAM** **FOR PREDICTING PRICE OF COIN**

An Activity Diagram is a type of behavioral diagram used in Unified Modeling Language (UML) to represent the flow of control or data through the system as seen in Fig. 3.4.3.1. It focuses on the flow of activities and actions, capturing the sequence of steps in a particular process or workflow. Activity diagrams are commonly used to model business processes, workflows, or any sequential activities in a system.

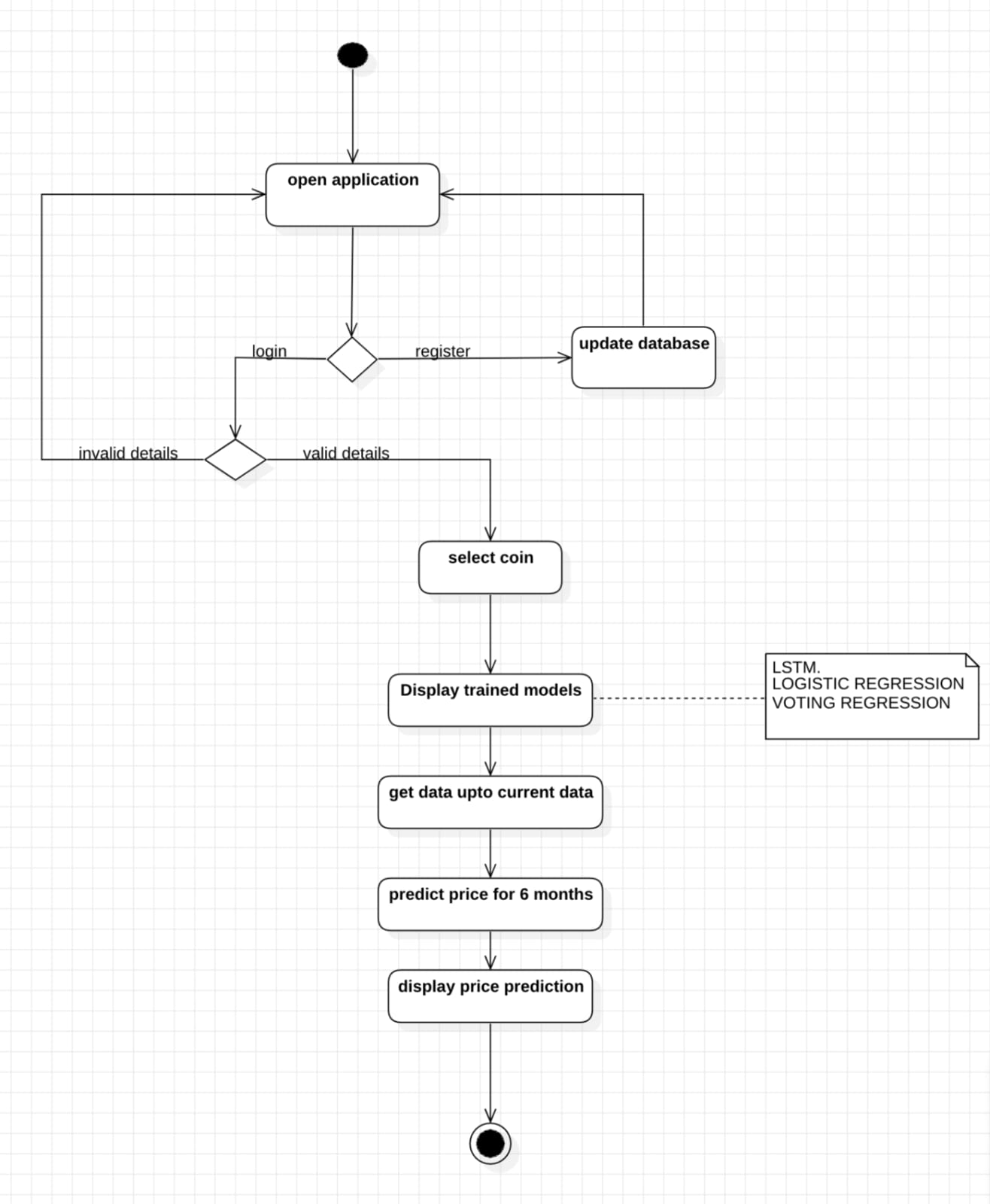


Fig. 3.4.3.1 Activity Diagram

**Flow Explanation**

* **Open Application**

The user starts by opening the application.

* **Login or Register**
  + The user is presented with the option to login or register:
    - **Login**: If the user already has an account, they log in by providing their credentials.
    - **Register**: New users must register, which updates the database with their details.
* **Validation**
  + If login details are valid, the user proceeds.
  + If login details are invalid, they are redirected back to the login page.
* **Select Coin**
  + The user selects a cryptocurrency (e.g., Bitcoin or Ethereum) for price prediction.
* **Get Data Up to Current Date**
  + Historical and up-to-date cryptocurrency data is retrieved for analysis.
* **Predict Price for 6 Months**
  + The system predicts the price of the selected cryptocurrency for the next six months using the trained models.
* **Display Price Prediction**
  + The predicted prices are displayed to the user, often as graphs or numerical outputs.
* **End of Workflow**
  + The process concludes, and the user can either close the application or perform additional predictions.

**3.4.4 SEQUENCE DIAGRAM**

A sequence diagram illustrates the flow of interactions between actors and system components over time as seen in Fig. 3.4.4.1, emphasizing the order in which messages are exchanged to achieve specific functionalities. Actors represent external entities that interact with the system, while lifelines depict the system components involved in the process. Messages are shown as arrows, indicating the flow of information or actions between these elements. By providing a step-by-step view of workflows, sequence diagrams help in understanding and designing the dynamic behaviour of a system.

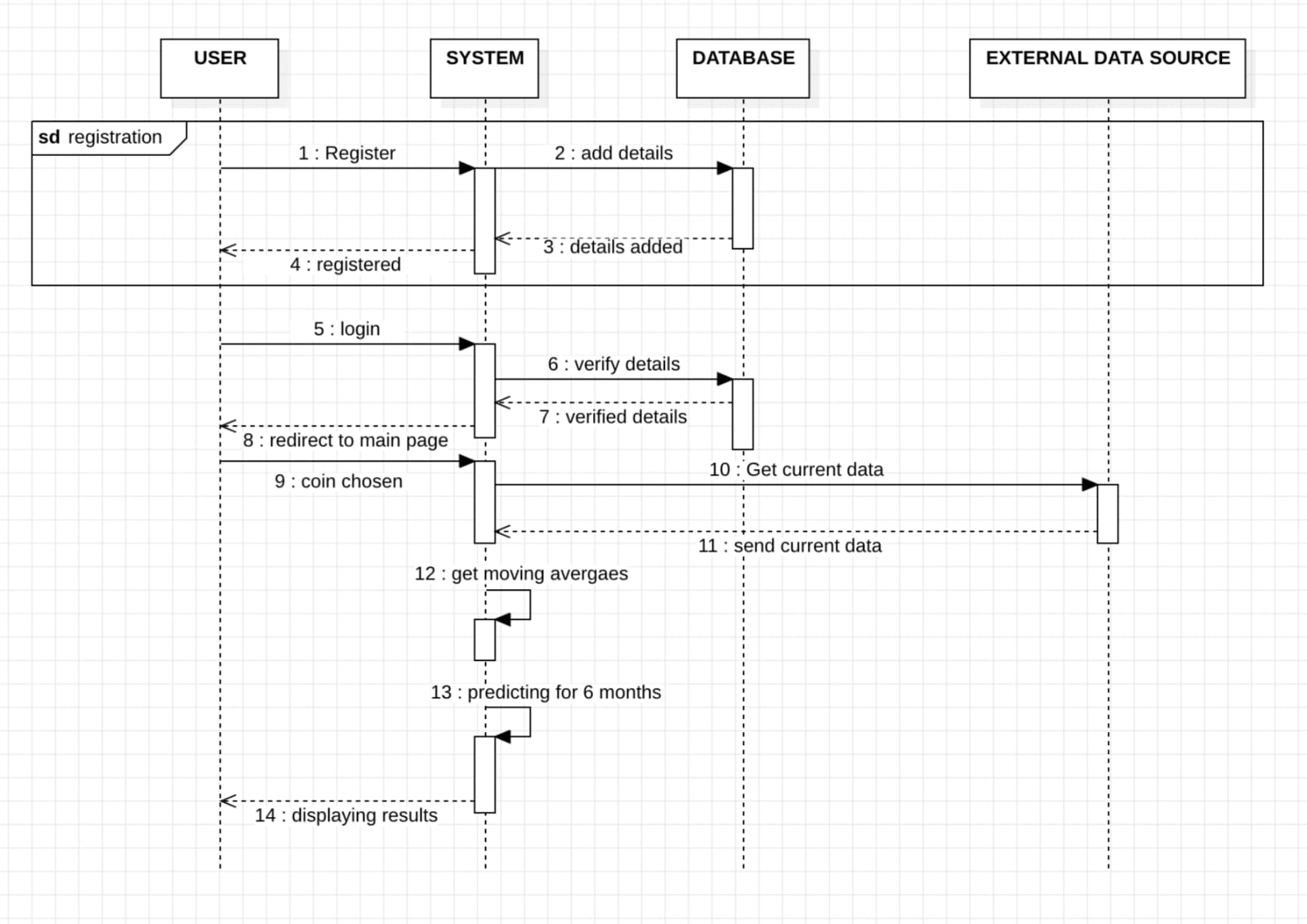


Fig. 3.4.4.1 Sequence Diagram

**Key Interactions and Relationships**

* **User and System**:
  + **Registration and Login**: The user registers and logs into the system using their credentials.
  + **Selecting Coin**: The user chooses which cryptocurrency (Bitcoin or Ethereum) they want predictions for.
* **System and Database**:
  + **Storing User Details**: When the user registers, the system stores their information in the database.
  + **Verifying Credentials**: During login, the system checks the user's credentials against the database to authenticate the user.
* **System and External Data Source**:
  + **Fetching Current Data**: To provide accurate predictions, the system requests the latest market data for the selected cryptocurrency from an external data source, such as a financial API.
* **Predictive Models**:
  + **Data Processing and Analysis**: The system processes the fetched data and feeds it into predictive models (e.g., LSTM, Logistic Regression, Voting Regressor) to forecast future prices.
* **System and User**:
  + **Displaying Results**: The system visualizes the predicted prices and other relevant information, presenting the results to the user through a user-friendly interface.

**3.4.5 COMPONENT DIAGRAM**

A Component Diagram is a type of structural diagram used in software engineering to represent the components of a system and how they interact or depend on each other. It shows how the components (which could be software modules, subsystems, or other significant parts) are organized and connected within a system. In this diagram, each component encapsulates a set of related functionalities and interfaces as shown in Fig. 3.4.5.1.

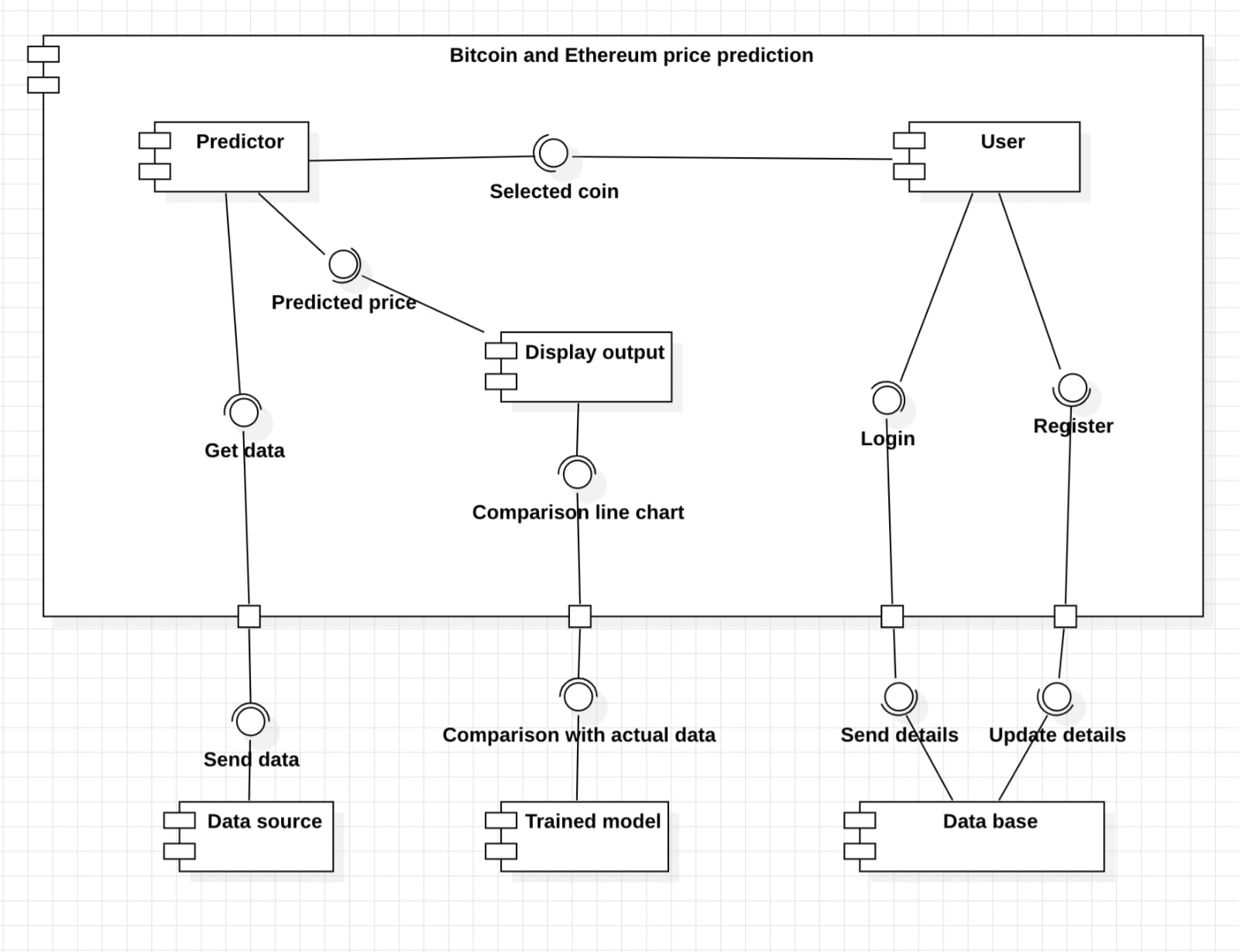


Fig. 3.4.5.1 Component Diagram

**Main Components:**

1. **Predictor**: This component is responsible for predicting the prices of Bitcoin and Ethereum. It gets data from the Data Source component and sends the predicted prices to the Display Output component.
2. **User**: This component allows users to interact with the system. Users can register, log in, and update their details. The user details are sent to and updated in the Database component.
3. **Display Output**: This component displays the predicted prices and comparison line charts. It receives the selected coin and predicted prices from the Predictor component and compares them with actual data from the Trained Models component.
4. **Data Source**: This component sends data to the Predictor component for price prediction.
5. **Trained Models**: This component provides actual data for comparison with the predicted prices.
6. **Database**: This component stores user details and updates them as needed.

**3.5 METHODOLOGY**

**Data Acquisition**

Live data is obtained from Yahoo Finance using libraries like yfinance. This includes both real-time cryptocurrency price updates and extensive historical data. The historical data provides a foundation for training the models, while live data ensures that predictions are relevant and up-to-date.

**Model Steps**

**Data:** Historical data is used for training to showcase model performance and enable comparison.

**Time Frame Generation:** Data is split into time frames (e.g., 7 days, 30 days) to allow models to predict prices over varying periods, ensuring flexibility in forecasting multiple intervals at once.

**Models Used**

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

LSTM is a type of Recurrent Neural Network (RNN), which is designed to process sequences of data. This makes LSTM ideal for time-series forecasting, where the goal is to predict future values based on past data. In the case of cryptocurrency price prediction, LSTM is employed to analyse the historical price movements of Bitcoin and Ethereum to capture any long-term trends or seasonal patterns.

Unlike standard RNNs, LSTM networks have a specialized architecture that allows them to store and manage information over long periods. This "memory" feature is crucial for learning long-term dependencies in sequential data, such as cryptocurrency prices, which can exhibit patterns that span months or even years. The LSTM can remember critical information from earlier time steps and use it to influence future predictions, making it effective in capturing long-term market trends.

**How it works:**

LSTM models are trained on a time-series dataset consisting of historical prices. As the model is exposed to more data, it learns to recognize recurring patterns and fluctuations in price movements. Once trained, the LSTM can predict future prices by analysing past price trends, trading volumes, and other relevant factors that contribute to price movements.

**Why it’s used:**

LSTM is particularly effective for long-term predictions because it can detect patterns that are not immediately obvious but are critical for accurate forecasting. In the cryptocurrency market, where price trends can change gradually over time due to factors like market sentiment, regulations, and technological advancements, LSTM’s ability to capture long-term dependencies provides a significant advantage over simpler models.

**Linear Regression**

Linear Regression is a statistical model that is typically used for predicting continuous numerical outcomes. In this system, it is adapted for forecasting the actual future price of a cryptocurrency over a given time period (e.g., the next day, week, or month).

**How it works:**

Linear Regression uses input features such as past prices, market volume, and other technical indicators to estimate the future price of a cryptocurrency. The output is a continuous numerical value representing the predicted price. The model fits a linear equation to the historical data, minimizing the difference between the predicted and actual prices.

**Why it’s used:**

Linear Regression is particularly useful for generating straightforward and interpretable predictions about future prices. While it does not indicate the direction separately, it provides an exact forecast of the price value, helping with more detailed investment decisions. The model’s simplicity, speed, and ease of interpretation make it an excellent choice for short-term price forecasting tasks.

**Voting Regressor**

The Voting Regressor is an ensemble learning technique that combines the predictions of multiple different regression models to improve the overall prediction accuracy and robustness. Instead of relying on a single model, the ensemble aggregates outputs from several models, aiming to balance their individual strengths and weaknesses.

In this system, the Voting Regressor uses three base models:

* Random Forest Regressor
* Decision Tree Regressor
* Gradient Boosting Regressor

The final prediction is the average of the individual predictions from these models.

**How it works:**

The Voting Regressor aggregates the predictions of its base by averaging their outputs. Each model makes its own prediction, and the final output is the average of these predictions. This helps mitigate the risk of individual models overfitting or underperforming in certain situations. For example, if one model predicts a much higher or lower price than the others, the averaging process helps to smooth out these extremes and produce a more balanced forecast.

**Random Forest Regressor in Voting Regressor:**

* A Random Forest is an ensemble of multiple Decision Trees trained on different random subsets of the data and features (a method known as "bagging").
* It aggregates the predictions of individual trees to reduce overfitting and variance.

Random Forest prediction formula (average of tree predictions):

where:

* T is the number of trees,
* Yt is the prediction of the tth tree.

Strength: Captures complex nonlinear patterns, very robust against overfitting (compared to single trees).

**Decision Tree Regressor in Voting Regressor:**

* A Decision Tree Regressor splits the data into branches based on feature thresholds to predict continuous output values.
* It models nonlinear relationships well but is prone to overfitting if not properly controlled.

Decision rule at each split:

where:

* xj is a feature,
* s is a threshold value.

Strengths: Easy to interpret, can capture sharp changes in data trends.

**Gradient Boosting Regressor in Voting Regressor:**

* Gradient **Boosting** builds models **sequentially**, where each new model tries to correct the errors made by previous ones.
* It minimizes a loss function (such as Mean Squared Error) by **adding weak learners** step-by-step.

Gradient Boosting update formula:

where:

* Fm(x) is the ensemble prediction after mmm steps,
* hm(x) is the new weak learner (e.g., a small decision tree),
* γm is the learning rate.

Strengths: Excellent predictive performance, handles complex patterns very well.

**Why it’s used:**

The Voting Regressor is chosen because it leverages the diverse strengths of different algorithms:

* Random Forest offers stability and resistance to overfitting.
* Decision Trees capture simple and intuitive splits in data.
* Gradient Boosting efficiently corrects errors for enhanced accuracy.

By averaging their outputs, the ensemble:

* Reduces variance (errors from model sensitivity to data fluctuations),
* Reduces bias (errors from overly simplistic models),
* Improves generalization to unseen data.

Thus, the final prediction is more accurate, stable, and robust compared to any single base model.

**Predicting Prices**

The system uses live data to further train the Voting Regressor model. This model predicts up to six months of future price data for the selected coin (Bitcoin or Ethereum). Predictions for incremental milestones, such as next day, next week, and next month prices, are also extracted.

**4. CODE AND IMPLEMENTATION**

**4.1 CODE**

**Main.py**

from flask import Flask, render\_template, request, redirect, url\_for, session,send\_from\_directory

import yfinance as yf

import os

import pymysql

import pandas as pd #pandas to read and explore dataset

import numpy as np

import io

import base64

import random

from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import mean\_squared\_error

from sklearn.ensemble import RandomForestRegressor, VotingRegressor

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import GradientBoostingRegressor

from datetime import date, timedelta

app = Flask(\_\_name\_\_)

app.secret\_key = 'welcome'

def train\_model(symbol):

end\_date = date.today()

start\_date = end\_date - timedelta(days=730)

df = yf.download(symbol, start=start\_date, end=end\_date)

df = df.reset\_index()

df['year'] = df['Date'].dt.year

df['month'] = df['Date'].dt.month

df['day'] = df['Date'].dt.day

X = df[['year', 'month', 'day']]

Y = df['Close'].values.reshape(-1, 1)

features\_scaler = MinMaxScaler()

target\_scaler = MinMaxScaler()

X\_scaled = features\_scaler.fit\_transform(X)

Y\_scaled = target\_scaler.fit\_transform(Y)

rf = RandomForestRegressor()

dt = DecisionTreeRegressor()

gb = GradientBoostingRegressor()

voting = VotingRegressor(estimators=[('rf', rf), ('dt', dt), ('gb', gb)])

voting.fit(X\_scaled, Y\_scaled)

return voting, features\_scaler, target\_scaler

@app.route('/PredictPricesAction', methods=['GET', 'POST'])

def PredictPricesAction():

if request.method == 'POST':

import yfinance as yf

from datetime import date, timedelta

symbol = request.form['t1'].strip()

num\_days = int(request.form['t2'].strip())

# Map user-friendly name to ticker

ticker\_map = {

'Bitcoin': 'BTC-USD',

'ETH': 'ETH-USD'

}

ticker = ticker\_map.get(symbol)

if not ticker:

return render\_template('UserScreen.html', data="Invalid symbol selected.")

# Fetch 1 year of historical data from today

end\_date = date.today()

start\_date = end\_date - timedelta(days=365)

df = yf.download(ticker, start=start\_date, end=end\_date)

if df.empty:

return render\_template('UserScreen.html', data="Failed to fetch data for selected symbol.")

df = df.reset\_index()

df['year'] = df['Date'].dt.year

df['month'] = df['Date'].dt.month

df['day'] = df['Date'].dt.day

# Prepare dataset

X = df[['year', 'month', 'day']]

Y = df['Close'].values.reshape(-1, 1)

# Normalize

features\_scaler = MinMaxScaler()

target\_scaler = MinMaxScaler()

X\_scaled = features\_scaler.fit\_transform(X)

Y\_scaled = target\_scaler.fit\_transform(Y)

# Train model

rf = RandomForestRegressor()

dt = DecisionTreeRegressor()

gb = GradientBoostingRegressor()

voting = VotingRegressor(estimators=[('rf', rf), ('dt', dt), ('gb', gb)])

voting.fit(X\_scaled, Y\_scaled)

# Generate future dates

future\_data = []

current\_date = date.today()

for i in range(num\_days):

future = current\_date + timedelta(days=i+1)

future\_data.append([future.year, future.month, future.day])

future\_scaled = features\_scaler.transform(future\_data)

predictions\_scaled = voting.predict(future\_scaled)

predictions = target\_scaler.inverse\_transform(predictions\_scaled.reshape(-1, 1))

# Render predictions in HTML

output = '<table border="1" align="center" width="100%" style="text-align: center;">'

output += '<tr><th>Day No</th><th>Future Date</th><th>Predicted Price (USD)</th></tr>'

for i, price in enumerate(predictions):

future = current\_date + timedelta(days=i+1)

output += f'<tr><td>{i+1}</td><td>{future}</td><td>{round(price[0], 2)}</td></tr>'

output += '</table><br><br>'

labels = [str(current\_date + timedelta(days=i+1)) for i in range(num\_days)]

prices = [round(p[0], 2) for p in predictions]

return render\_template('UserScreen.html', data=output, labels=labels, prices=prices, coin=symbol)

@app.route('/RegisterAction', methods=['GET', 'POST'])

def RegisterAction():

if request.method == 'POST':

username = request.form['t1']

password = request.form['t2']

contact = request.form['t3']

email = request.form['t4']

address = request.form['t5']

command = 'not\_found'

db\_connect = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'bitcoin',charset='utf8')

with db\_connect:

cursor = db\_connect.cursor()

cursor.execute("select username from signup where username = '"+username+"'")

rows = cursor.fetchall()

for row in rows:

if row[0] == username:

command = username+' Given Username already exists'

break

if command == 'not\_found':

db\_connect = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'bitcoin',charset='utf8')

cursor = db\_connect.cursor()

query = "INSERT INTO signup VALUES('"+username+"','"+password+"','"+contact+"','"+email+"','"+address+"')"

cursor.execute(query)

db\_connect.commit()

if cursor.rowcount == 1:

command = 'New User Registration Task Completed'

return render\_template('Register.html', data=command)

@app.route('/UserLoginAction', methods=['GET', 'POST'])

def UserLoginAction():

if request.method == 'POST':

global user

user = request.form['t1']

password = request.form['t2']

command = 'none'

db\_connect = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'bitcoin',charset='utf8')

with db\_connect:

cursor = db\_connect.cursor()

cursor.execute("select username, password from signup")

rows = cursor.fetchall()

for row in rows:

if row[0] == user and row[1] == password:

command = 'success'

break

if command == "success":

return render\_template('UserScreen.html', data="Welcome "+user)

else:

return render\_template('UserLogin.html', data="Invalid login details")

@app.route('/PredictPrices', methods=['GET', 'POST'])

def PredictPrices():

return render\_template('PredictPrices.html', data='')

@app.route('/Register', methods=['GET', 'POST'])

def Register():

return render\_template('Register.html', data='')

@app.route('/UserLogin', methods=['GET', 'POST'])

def UserLogin():

return render\_template('UserLogin.html', data='')

@app.route('/', methods=['GET', 'POST'])

def indexx():

return render\_template('index.html', data='')

@app.route('/index', methods=['GET', 'POST'])

def index():

return render\_template('index.html', data='')

@app.route('/Logout')

def Logout():

return render\_template('index.html', data='')

if \_\_name\_\_ == '\_\_main\_\_':

app.run()

**index.html**

<html>

<head>

<title>Ethereum & Bitcoin Price Prediction</title>

<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">

</head>

<style>

body {

  background-image: url({{ url\_for('static', filename='images/back.jpg') }});

}

</style>

<body>

<div id="wrapper">

    <div id="header">

        <div id="logo">

            <center><font size="4" color="white">Bitcoin & Ethereum Price Predictor</font></center></font>

        </div>

        <div id="slogan">

        </div>

    </div>

    <div id="menu">

        <ul><center>

      <li><a href="index"><font size="" color="black">Home</a></font></li>

      <li><a href="UserLogin"><font size="" color="black">User Login</a></li>

     <li><a href="Register"><font size="" color="black">New User Registration</a></li>

        </center></ul>

        <br class="clearfix" />

    </div>

    <div id="splash">

        <img class="pic" src={{ url\_for('static', filename='images/investor.jpg') }} width="870" height="230" alt="" />

    </div>

    <br/>

    <font size="3" color="black"><center>{{ data|safe }}</center></font><br/>

        <img src="data:image/png;base64, {{ img }}" alt="" />

</body>

</html>

**PredictPrices.html**

<html>

<head>

<title>Ethereum & Bitcoin Price Prediction</title>

<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">

 <script LANGUAGE="Javascript" >

function validate(){

    var x=document.forms["f1"]["t1"].value;

    var y=document.forms["f1"]["t2"].value;

    if(x == null || x==""){

        window.alert("Username must be enter");

        document.f1.username.focus();

        return false;

    }

    if(y == null || y==""){

        window.alert("Password must be enter");

        document.f1.password.focus();

        return false;

    }

    return true;

}

</script>

</head>

<style>

body {

  background-image: url({{ url\_for('static', filename='images/back.jpg') }});

}

</style>

<body>

<div id="wrapper">

    <div id="header">

        <div id="logo">

            <center><font size="4" color="white">Bitcoin & Ethereum Price Predictor</font></center></font>

        </div>

        <div id="slogan">

        </div>

    </div>

    <div id="menu">

        <ul><center>

      <ul><center>

        <li><a href="PredictPrices"><font size="" color="black">Forecast Stock Prices</font></a></font></li>

         <li><a href="index"><font size="" color="black">Logout</font></a></font></li>

        </center></ul>

<br class="clearfix" />

                </div>

    <div id="splash">

        <img class="pic" src={{ url\_for('static', filename='images/investor.jpg') }} width="870" height="230" alt="" />

    </div>

            <center>

<form name="f1" method="post" action="/PredictPricesAction" OnSubmit="return validate()">

    <br/>

   <h3 style="text-align: center; color: #333; margin-bottom: 20px;">

    <b>Future Stock Prices Forecasting Screen</b>

</h3>

<div style="text-align: center; color: black; margin-bottom: 25px;">

    {{ data }}

</div>

<table align="center" style="

    background-color: #f9f9f9;

    border-radius: 12px;

    box-shadow: 0 4px 8px rgba(0,0,0,0.1);

    padding: 30px 40px;

">

    <tr>

        <td style="padding: 15px; font-weight: bold; color: #333;">Choose Stock Symbol</td>

        <td style="padding: 15px;">

            <select name="t1" style="padding: 8px 12px; font-family: Comic Sans MS; border-radius: 6px; border: 1px solid #ccc; width: 180px;">

                <option value="Bitcoin">Bitcoin</option>

                <option value="ETH">ETH</option>

            </select>

        </td>

    </tr>

    <tr>

        <td style="padding: 15px; font-weight: bold; color: #333;">Choose Forecast Tenure</td>

        <td style="padding: 15px;">

            <select name="t2" style="padding: 8px 12px; font-family: Comic Sans MS; border-radius: 6px; border: 1px solid #ccc; width: 180px;">

                <option value="7">1 Week</option>

                <option value="30">1 Month</option>

                <option value="90">3 Months</option>

                <option value="180">6 Months</option>

            </select>

        </td>

    </tr>

    <tr>

        <td colspan="2" style="text-align: center; padding: 25px;">

            <input type="submit" value="Predict" style="padding: 12px 24px; background-color: #4CAF50; color: white; border: none; border-radius: 6px; cursor: pointer; font-size: 16px;">

        </td>

    </tr>

</table>

            <br/><br/><br/><br/>

                </div>

                </div>

    </body>

</html>

**Register.html**

<html>

<head>

<title>Ethereum & Bitcoin Price Prediction</title>

<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">

 <script LANGUAGE="Javascript" >

function validate(){

    var x=document.forms["f1"]["t1"].value;

    var y=document.forms["f1"]["t2"].value;

    var a=document.forms["f1"]["t3"].value;

    var b=document.forms["f1"]["t4"].value;

    var c=document.forms["f1"]["t5"].value;

    if(x == null || x==""){

        window.alert("Username must be entered");

        document.f1.t1.focus();

        return false;

    }

    if(y == null || y==""){

        window.alert("Password must be entered");

        document.f1.t2.focus();

        return false;

    }

    if(a == null || a==""){

        window.alert("Phone No must be entered");

        document.f1.t3.focus();

        return false;

    }

    if(isNaN(a)){

        alert("Contact No must be numeric");

        document.f1.t3.focus();

        return false;

    }

    if(b == null || b==""){

        window.alert("Email ID must be entered");

        document.f1.t4.focus();

        return false;

    }

    var filter = /^([a-zA-Z0-9\_\.\-])+\@([a-z]+\.)+(com)+$/;

    if (!filter.test(b)) {

        window.alert('Please enter valid email address');

        document.f1.t4.focus();

        return false;

    }

    if(c == null || c==""){

        window.alert("Address must be enter");

        document.f1.t5.focus();

        return false;

    }

    return true;

}

</script>

</head>

<style>

body {

  background-image: url({{ url\_for('static', filename='images/back.jpg') }});

}

</style>

<body>

<div id="wrapper">

    <div id="header">

        <div id="logo">

            <center><font size="4" color="white">Bitcoin & Ethereum Price Predictor</font></center></font>

        </div>

        <div id="slogan">

        </div>

    </div>

    <div id="menu">

        <ul><center>

      <ul><center>

      <li><a href="index"><font size="" color="black">Home</a></font></li>

      <li><a href="UserLogin"><font size="" color="black">User Login</a></li>

     <li><a href="Register"><font size="" color="black">New User Registration</a></li>

        </center></ul>

<br class="clearfix" />

                </div>

    <div id="splash">

        <img class="pic" src={{ url\_for('static', filename='images/investor.jpg') }} width="870" height="230" alt="" />

    </div>

            <center>

<form name="f1" method="post" action="/RegisterAction" OnSubmit="return validate()">

    <br/>

   <h3 style="text-align: center; color: #333; margin-bottom: 20px;"><b>User Registration Screen</b></h3>

<div style="text-align: center; color: black; margin-bottom: 25px;">

    {{ data }}

</div>

<table align="center" style="

    background-color: #f9f9f9;

    border-radius: 12px;

    box-shadow: 0 4px 8px rgba(0,0,0,0.1);

    padding: 30px 40px;

">

    <tr>

        <td style="padding: 15px; color: #333; font-weight: bold;">Username</td>

        <td style="padding: 15px;">

            <input type="text" name="t1" style="font-family: Comic Sans MS; width: 250px; padding: 10px; border-radius: 6px; border: 1px solid #ccc;" required>

        </td>

    </tr>

    <tr>

        <td style="padding: 15px; color: #333; font-weight: bold;">Password</td>

        <td style="padding: 15px;">

            <input type="password" name="t2" style="font-family: Comic Sans MS; width: 250px; padding: 10px; border-radius: 6px; border: 1px solid #ccc;" required>

        </td>

    </tr>

    <tr>

        <td style="padding: 15px; color: #333; font-weight: bold;">Contact No</td>

        <td style="padding: 15px;">

            <input type="text" name="t3" style="font-family: Comic Sans MS; width: 180px; padding: 10px; border-radius: 6px; border: 1px solid #ccc;" required>

        </td>

    </tr>

    <tr>

        <td style="padding: 15px; color: #333; font-weight: bold;">Email ID</td>

        <td style="padding: 15px;">

            <input type="text" name="t4" style="font-family: Comic Sans MS; width: 300px; padding: 10px; border-radius: 6px; border: 1px solid #ccc;" required>

        </td>

    </tr>

    <tr>

        <td style="padding: 15px; color: #333; font-weight: bold;">Address</td>

        <td style="padding: 15px;">

            <input type="text" name="t5" style="font-family: Comic Sans MS; width: 400px; padding: 10px; border-radius: 6px; border: 1px solid #ccc;" required>

        </td>

    </tr>

    <tr>

        <td colspan="2" style="text-align: center; padding: 25px;">

            <input type="submit" value="Submit" style="padding: 12px 24px; background-color: #4CAF50; color: white; border: none; border-radius: 6px; cursor: pointer; font-size: 16px;">

        </td>

    </tr>

</table>

            <br/><br/><br/><br/>

                </div>

                </div>

    </body>

</html>

**UserLogin.html**

<html>

<head>

<title>Ethereum & Bitcoin Price Prediction</title>

<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">

 <script LANGUAGE="Javascript" >

function validate(){

    var x=document.forms["f1"]["t1"].value;

    var y=document.forms["f1"]["t2"].value;

    if(x == null || x==""){

        window.alert("Username must be entered");

        document.f1.username.focus();

        return false;

    }

    if(y == null || y==""){

        window.alert("Password must be entered");

        document.f1.password.focus();

        return false;

    }

    return true;

}

</script>

</head>

<style>

body {

  background-image: url({{ url\_for('static', filename='images/back.jpg') }});

}

</style>

<body>

<div id="wrapper">

    <div id="header">

        <div id="logo">

            <center><font size="4" color="white">Bitcoin & Ethereum Price Predictor</font></center></font>

        </div>

        <div id="slogan">

        </div>

    </div>

    <div id="menu">

        <ul><center>

      <ul><center>

      <li><a href="index"><font size="" color="black">Home</a></font></li>

      <li><a href="UserLogin"><font size="" color="black">User Login</a></li>

     <li><a href="Register"><font size="" color="black">New User Registration</a></li>

        </center></ul>

<br class="clearfix" />

                </div>

    <div id="splash">

        <img class="pic" src={{ url\_for('static', filename='images/investor.jpg') }} width="870" height="230" alt="" />

    </div>

            <center>

<form name="f1" method="post" action="/UserLoginAction" OnSubmit="return validate()">

    <br/>

   <h3 style="text-align: center; color: #333; margin-bottom: 20px;"><b>User Login Screen</b></h3>

<div style="text-align: center; color: black; margin-bottom: 25px;">

    {{ data }}

</div>

<table align="center" style="

    background-color: #f9f9f9;

    border-radius: 12px;

    box-shadow: 0 4px 8px rgba(0,0,0,0.1);

    padding: 30px 40px;

">

    <tr>

        <td style="padding: 15px; color: #333; font-weight: bold;">Username</td>

        <td style="padding: 15px;">

            <input type="text" name="t1" style="font-family: Comic Sans MS; width: 250px; padding: 10px; border-radius: 6px; border: 1px solid #ccc;" required>

        </td>

    </tr>

    <tr>

        <td style="padding: 15px; color: #333; font-weight: bold;">Password</td>

        <td style="padding: 15px;">

            <input type="password" name="t2" style="font-family: Comic Sans MS; width: 250px; padding: 10px; border-radius: 6px; border: 1px solid #ccc;" required>

        </td>

    </tr>

    <tr>

        <td colspan="2" style="text-align: center; padding: 20px;">

            <input type="submit" value="Login" style="padding: 12px 24px; background-color: #4CAF50; color: white; border: none; border-radius: 6px; cursor: pointer; font-size: 16px;">

        </td>

    </tr>

</table>

            <br/><br/><br/><br/>

                </div>

                </div>

    </body>

</html>

**UserScreen.html**

<html>

<head>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<script src="https://cdn.jsdelivr.net/npm/chartjs-plugin-zoom@1.2.1/dist/chartjs-plugin-zoom.min.js"></script>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<title>Ethereum & Bitcoin Price Predictor</title>

<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">

</head>

<style>

body {

  background-image: url({{ url\_for('static', filename='images/back.jpg') }});

}

</style>

<body>

<div id="wrapper">

    <div id="header">

        <div id="logo">

            <center><font size="4" color="white">Bitcoin & Ethereum Price Predictor</font></center></font>

        </div>

        <div id="slogan">

        </div>

    </div>

    <div id="menu">

        <ul><center>

        <li><a href="PredictPrices"><font size="" color="black">Forecast Stock Prices</font></a></font></li>

         <li><a href="index"><font size="" color="black">Logout</font></a></font></li>

        </center></ul>

        <br class="clearfix" />

    </div>

    <div id="splash">

        <img class="pic" src={{ url\_for('static', filename='images/investor.jpg') }} width="870" height="230" alt="" />

    </div>

{% if labels and prices %}

    <div style="width: 80%; margin: auto;">

    <canvas id="priceChart"></canvas>

</div>

<canvas id="myChart"></canvas>

<script>

    const labels = {{ labels | tojson }};

    const prices = {{ prices | tojson }};

    const ctx = document.getElementById('myChart').getContext('2d');

    const chart = new Chart(ctx, {

        type: 'line',

        data: {

            labels: labels,

            datasets: [{

                label: '{{ coin }} Price Prediction',

                data: prices,

                fill: false,

                borderColor: 'rgb(75, 192, 192)',

                tension: 0.3,

                pointRadius: 3,

                pointHoverRadius: 6

            }]

        },

        options: {

            responsive: true,

            plugins: {

                title: {

                    display: true,

                    text: '{{ coin }} Price Prediction Over Time',

                    font: { size: 18 }

                },

                tooltip: {

                    mode: 'index',

                    intersect: false,

                    callbacks: {

                        label: function(context) {

                            return 'Price: $' + context.raw.toFixed(2);

                        }

                    }

                },

                zoom: {

                    zoom: {

                        wheel: { enabled: true },

                        pinch: { enabled: true },

                        mode: 'x',

                    },

                    pan: {

                        enabled: true,

                        mode: 'x',

                    }

                }

            },

            scales: {

                x: {

                    title: {

                        display: true,

                        text: 'Date'

                    }

                },

                y: {

                    title: {

                        display: true,

                        text: 'Price (USD)'

                    }

                }

            }

        }

    });

</script>

{% endif %}

    <div style="text-align: center; margin-top: 20px; padding: 20px; background: #f0f8ff; border-radius: 10px; width: 80%; margin-left: auto; margin-right: auto; font-size: 18px; font-weight: bold; color: #333; overflow-x: auto;">

    {{ data|safe }}

</div>

        <img src="data:image/png;base64, {{ img }}" alt="" />

</body>

</html>

**style.css**

/\*

Design by Free CSS Templates

http://www.freecsstemplates.org

Released for free under a Creative Commons Attribution 3.0 License

Name       : Big Business

Description: A two-column, fixed-width design with a bright color scheme.

Version    : 1.0

Released   : 20120210

\*/

\* {

    margin: 0;

    padding: 0;

}

a {

    text-decoration: underline;

    color: #0F8C8C;

}

a:hover {

    text-decoration: none;

}

body {

    font-size: 11.5pt;

    color: #5C5B5B;

    line-height: 1.75em;

    background: url(images/back.jpg) repeat-x top left;

}

body,input {

    font-family: Georgia, serif;

}

strong {

    color: #2C2B2B;

}

br.clearfix {

    clear: both;

}

h1,h2,h3,h4 {

    font-weight: normal;

    letter-spacing: -1px;

}

h2 {

    font-size: 2.25em;

}

h2,h3,h4 {

    color: #2C2B2B;

    margin-bottom: 1em;

}

h3 {

    font-size: 1.75em;

}

h4 {

    font-size: 1.5em;

}

img.alignleft {

    margin: 5px 20px 20px 0;

    float: left;

}

img.aligntop {

    margin: 5px 0 20px 0;

}

img.pic {

    padding: 5px;

    border: solid 1px #D4D4D4;

}

p {

    margin-bottom: 1.5em;

}

ul {

    margin-bottom: 1.5em;

}

ul h4 {

    margin-bottom: 0.35em;

}

.box {

    overflow: hidden;

    margin-bottom: 1em;

}

.date {

    background: #6E6E6E;

    padding: 5px 6px 5px 6px;

    margin: 0 6px 0 0;

    color: #FFFFFF;

    font-size: 0.8em;

    border-radius: 2px;

}

#content {

    width: 665px;

    float: left;

    padding: 0;

}

#content-box1 {

    width: 320px;

    float: left;

}

#content-box2 {

    margin: 0 0 0 345px;

    width: 320px;

}

#footer {

    margin: 40px 0 120px 0;

    text-align: center;

    color: #8C8B8B;

}

#footer a {

    color: #8C8B8B;

}

#header {

    height: 75px;

    position: relative;

    background: #6E6E6E url(images/img03.jpg) top left no-repeat;

    padding: 45px;

    color: #FFFFFF;

    width: 888px;

    border: solid 1px #7E7E7E;

    border-top-left-radius: 5px;

    border-top-right-radius: 5px;

    overflow: hidden;

}

#logo {

    line-height: 160px;

    height: 160px;

    padding: 5px 0 0 0;

    position: absolute;

    top: 0;

    left: 45px;

}

#logo a {

    text-decoration: none;

    color: #FFFFFF;

    text-shadow: 0 1px 1px #3E3E3E;

}

#logo h1 {

    font-size: 2.25em;

}

#slogan {

    line-height: 160px;

    height: 160px;

    padding: 5px 0 0 0;

    position: absolute;

    right: 45px;

    top: 0;

}

#slogan h2 {

    color: #BEBEBE;

    font-size: 1.25em;

    text-shadow: 0 1px 1px #3E3E3E;

}

#menu {

    padding: 0 45px 0 45px;

    position: relative;

    background: url(images/img01.gif) repeat-x top left;

    margin: 0 0 0 0;

    height: 60px;

    line-height: 60px;

    width: 890px;

    border-top: solid 1px #5AD7D7;

    text-shadow: 0 1px 1px #007D7D;

}

#menu a {

    text-decoration: none;

    color: #FFFFFF;

    font-size: 1.25em;

    letter-spacing: -1px;

}

#menu ul {

    list-style: none;

}

#menu ul li {

    padding: 0 10px 0 10px;

    display: inline;

}

#menu ul li.first {

    padding-left: 0;

}

#page {

    padding: 45px 45px 15px 45px;

    position: relative;

    width: 890px;

    margin: 0;

}

#page .section-list {

    list-style: none;

    padding-left: 0;

}

#page .section-list li {

    clear: both;

    padding: 30px 0 30px 0;

}

#page ul {

    list-style: none;

}

#page ul li {

    border-top: solid 1px #D4D4D4;

    padding: 15px 0 15px 0;

}

#page ul li.first {

    padding-top: 0;

    border-top: 0;

}

#page-bottom {

    position: relative;

    margin: 0;

    background: #6E6E6E url(images/img03.jpg) top left no-repeat;

    border: solid 1px #7E7E7E;

    width: 888px;

    padding: 45px 45px 0 45px;

    color: #DCDCDC;

    border-bottom-left-radius: 5px;

    border-bottom-right-radius: 5px;

}

#page-bottom a {

    color: #F5F5F5;

}

#page-bottom h2, #page-bottom h3, #page-bottom h4 {

    color: #FFFFFF;

}

#page-bottom ul {

    list-style: none;

}

#page-bottom ul li {

    border-top: solid 1px #8F8F8F;

    padding: 15px 0 15px 0;

}

#page-bottom ul li.first {

    padding-top: 0;

    border-top: 0;

}

#page-bottom-content {

    width: 665px;

    float: left;

}

#page-bottom-sidebar {

    width: 200px;

    margin: 0 0 0 690px;

}

#search input.form-submit {

    margin-left: 1em;

    color: #FFFFFF;

    padding: 10px;

    background: #2FACAC;

    border: 0;

}

#search input.form-text {

    border: solid 1px #8F8F8F;

    padding: 10px;

}

#sidebar {

    width: 200px;

    padding: 0;

    margin: 0 0 0 690px;

}

#splash {

    margin: 0 0 0 0;

    height: 250px;

    position: relative;

    padding: 45px 45px 10px 45px;

    width: 890px;

}

#splash .pic {

    padding: 9px;

}

#wrapper {

    position: relative;

    width: 980px;

    margin: 75px auto 0 auto;

    background: #FFFFFF;

}

**4.2 IMPLEMENTATION**

Create a directory folder as following and copy relevant pieces of code into the required parts: -

Crypto Predictor/

│

├── main.py

│

├── static/

│ ├── images/

│ └── style.css

│

├── templates/

│ ├── index.html

│ ├── PredictPrices.html

│ ├── Register.html

│ ├── UserLogin.html

│ └── UserScreen.html

For images add any 3 images of your liking with names back (upper half background for navbar), img01(Background for title part of navbar), img03(background for items in navbar (should be GIF type)) and investor (Banner).

**Installing python packages**

Use the following command to install the required packages in terminal: -

pip install flask yfinance pymysql pandas numpy scikit-learn requests lxml html5lib

**Setting up SQL Server**

The general MySQL Installer download is available at

<https://dev.mysql.com/downloads/windows/installer/>.

The MySQL Installer application can install, upgrade, and manage most MySQL products, including MySQL Workbench. Managing all of your MySQL products, including Workbench, with [MySQL Installer](https://dev.mysql.com/downloads/windows/installer/) is the recommended approach. It handles all requirements and prerequisites, configurations, and upgrades.

When executing [MySQL Installer](https://dev.mysql.com/doc/refman/8.0/en/mysql-installer.html), choose MySQL Workbench as one of the products to install. It is selected by default, and essentially executes the standalone MSI Installer package

Install the latest version of it.

To install MySQL Workbench:

* From an account with Administrator or Power User privileges, right-click the MSI file and select the Install item from the pop-up menu, or double-click the file.
* In the Setup Type page, select either a Complete or Custom installation. To use all features of MySQL Workbench choose the Complete setup type.
* Unless you choose otherwise, MySQL Workbench is installed in C:\%PROGRAMFILES%\MySQL\MySQL Workbench 8.0 edition\_type\, where %PROGRAMFILES% is the default directory for programs for your locale. The %PROGRAMFILES% directory is defined as C:\Program Files\ on most systems.

Create the required table using the following commands after opening SQL workbench from start menu

create database bitcoin;

use bitcoin;

create table signup(username varchar(50) primary key,

password varchar(50),

contact\_no varchar(15),

email varchar(50),

address varchar(80));

**5. TESTING**

**INTRODUCTION TO TESTING**

Testing is a critical aspect of software development aimed at verifying, validating, and ensuring the quality and reliability of various software components. It helps to minimize risks and optimize resource utilization throughout the development lifecycle. Although testing can be applied at any stage, implementing it early allows for the identification and resolution of defects before they escalate. Testing involves evaluating the software under different conditions and environments to assess its functionality, performance, and other essential attributes. Various testing methodologies are employed depending on the nature and objectives of the software being developed. In this project, the testing phase ensures that every module functions effectively, thereby contributing to the system’s overall reliability and performance.

Testing was an integral part of this project, focusing on assessing the system’s ability to accurately predict cryptocurrency prices, specifically Ethereum and Bitcoin, over different time horizons—namely 1 week, 1 month, 3 months, and 6 months. The objective was to ensure that the price prediction system delivers consistent, accurate, and reliable results under diverse scenarios and dynamic market conditions.

The test cases were carefully designed to cover the key aspects of the system’s functionality, including:

* Accurate transition to required pages.
* Accurate prediction of Ethereum and Bitcoin prices across the specified future periods.
* Stability and robustness of predictions.
* Proper graph generation.

Each test case was executed systematically, and the results were thoroughly documented to compare actual outcomes against expected results.

**TEST CASES**

Table 5.1 Test Cases of Bitcoin and Ethereum Predictor

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S. No** | **Test Case Name** | **Input** | **Expected Output** | **Actual Output** | **Remarks** |
| 1 | Redirect after login | Name and Password | Redirects to /UserLoginAction. | Redirects to /UserLoginAction. | Pass |
| 2 | Clicking on Forecast stock prices | None | Redirects to /PredictPrices and displays options to select coin and forecast period. | Redirects to /PredictPrices and displays options to select coin and forecast period. | Pass |
| 3 | Clicking on Predict | Coin Symbol and Forecast Length | Redirects to /PredictPricesAction and shows the graph and table with predicted prices. | Redirects to /PredictPricesAction and shows the graph and table with predicted prices. | Pass |
| 4 | Clicking on Logout | None | Redirects to /Index. | Redirects to /Index. | Pass |
| 5 | Clicking on New User Registration | None | Redirects to /Register and displays registration form. | Redirects to /Register and displays registration form. | Pass |
| 6 | Clicking on User Login | None | Redirects to /UserLogin and displays the login form | Redirects to /UserLogin and displays the login form | Pass |

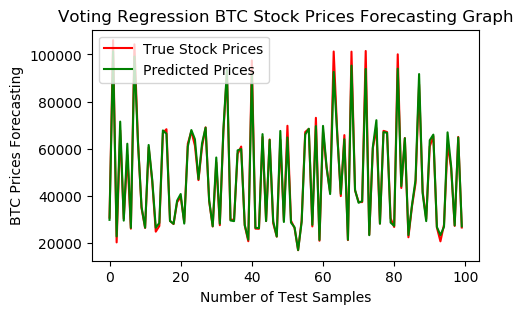
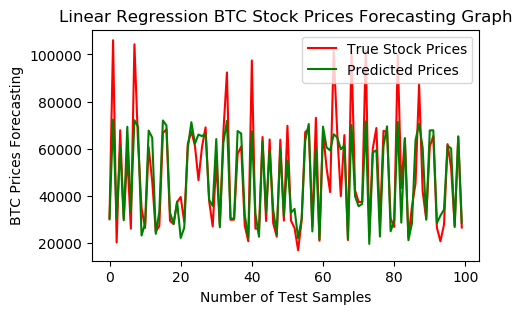
Table 5.1, shows the results of the test cases completed. It can be concluded that the system redirects properly and there are no issues in generated graphs and tables based on the selected coin and forecast period.

**6. RESULTS**

Table 6.1 Bitcoin and Ethereum Performance Comparison

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Algorithm** | **Bitcoin - 1 Year** | | | **Bitcoin - 2 Years** | | | **Ethereum - 1 Year** | | | **Ethereum - 2 Years** | | |
| **R2** | **RMSE** | **MAE** | **R2** | **RMSE** | **MAE** | **R2** | **RMSE** | **MAE** | **R2** | **RMSE** | **MAE** |
| Linear Regression | 0.4815 | 0.1598 | 0.1416 | 0.7556 | 0.1296 | 0.0849 | 0.0241 | 0.2920 | 0.2611 | 0.7390 | 0.1326 | 0.1016 |
| LSTM Regression | 0.9704 | 0.0382 | 0.0301 | 0.9467 | 0.0605 | 0.0331 | **0.9321** | **0.0752** | 0.0562 | 0.9224 | 0.0723 | 0.0460 |
| Voting Regression  (Proposed) | **0.9822** | **0.0296** | **0.0232** | **0.9898** | **0.0265** | **0.0167** | 0.9271 | 0.0779 | **0.0537** | **0.9749** | **0.0412** | **0.0262** |

From table 6.1 we can conclude that voting regressor is the best model for use due to its comparatively higher R2Scores, lower RMSE and lower MAE. It’s also better to take 2 years of data due to proper gap in performance between LSTM and voting regressor for Ethereum. The values that are the best comparatively between them are highlighted to show which model is giving better results



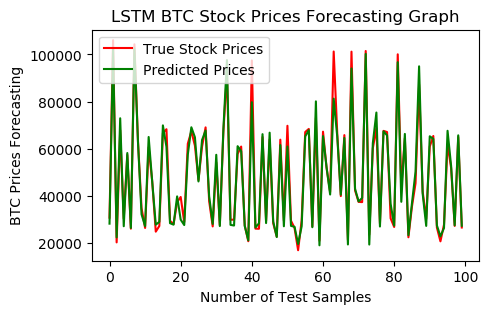
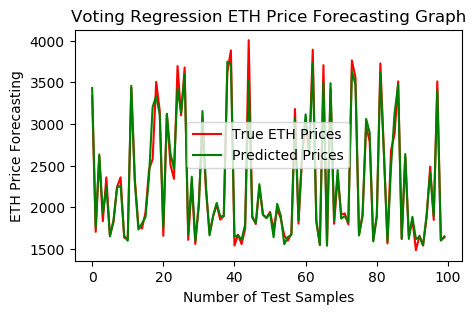
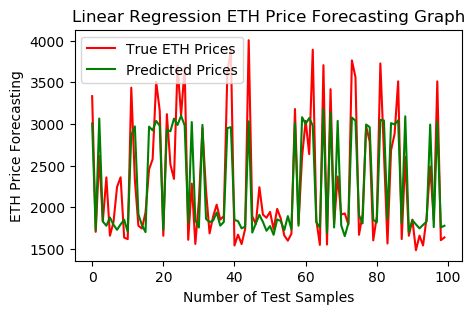


Fig. 6.1 Bitcoin Model Comparisons

Fig. 6.1 shows the output of bitcoin for 2 years of data used to predict prices. This is the visual representation of the predictions on which the metrics in Table 6.1 are based on.



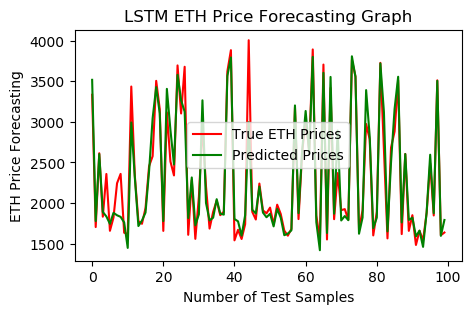


Fig. 6.2 Ethereum Model Comparisons

Fig. 6.2 shows the output of Ethereum for 2 years of data used to predict prices. This is the visual representation of the predictions on which the metrics in Table 6.1 are based on.

**Website Results**

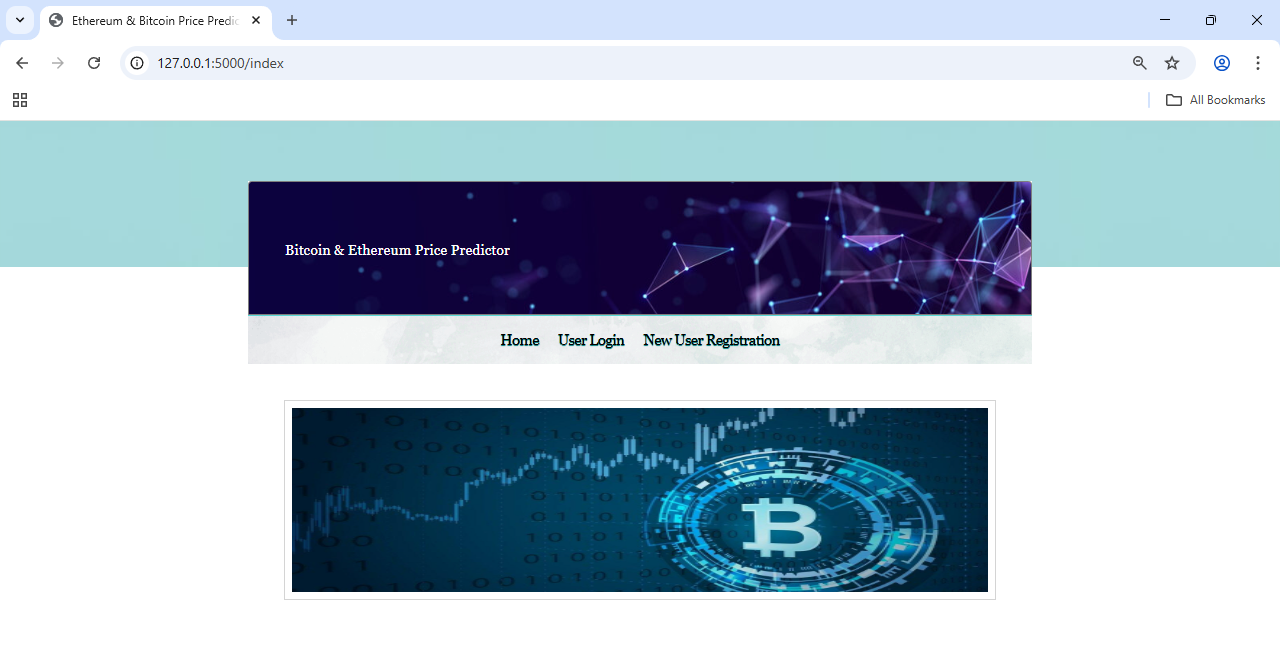


Fig. 6.3 Initial page

Fig. 6.3 shows the initial page displayed upon initially loading the local host address.

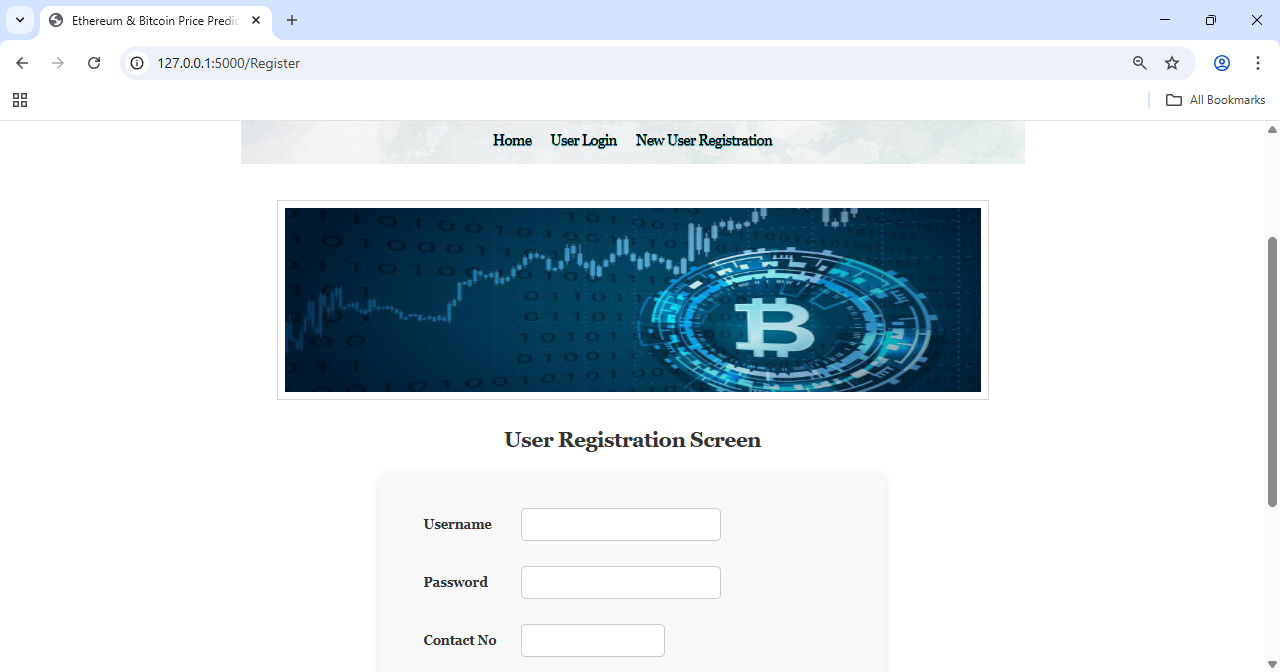


Fig. 6.4 Registration Page

Fig. 6.4 shows the registration page displayed clicking the New User Registration section of the navbar. It contains a form with name, password, contact no, email and address with a submit button.

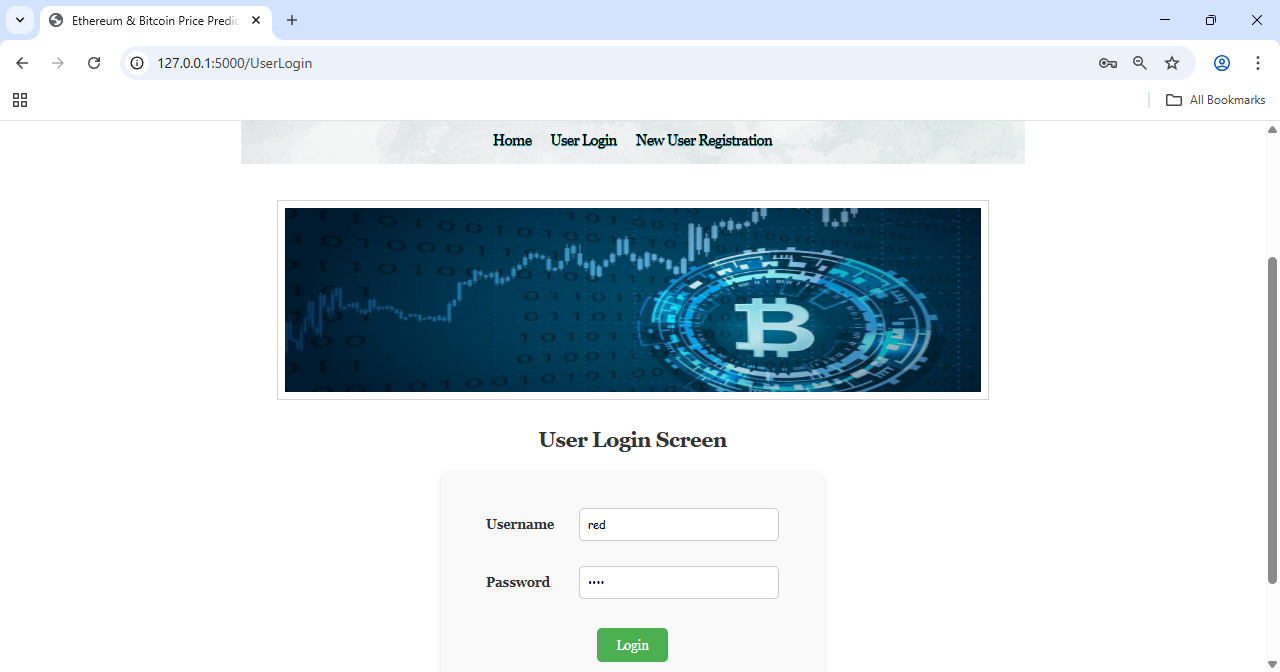


Fig. 6.5 Login page

Fig. 6.5 shows the Login page displayed upon clicking User Login in the navbar. It contains name and password fields to allow for login which redirects to main page of the project.

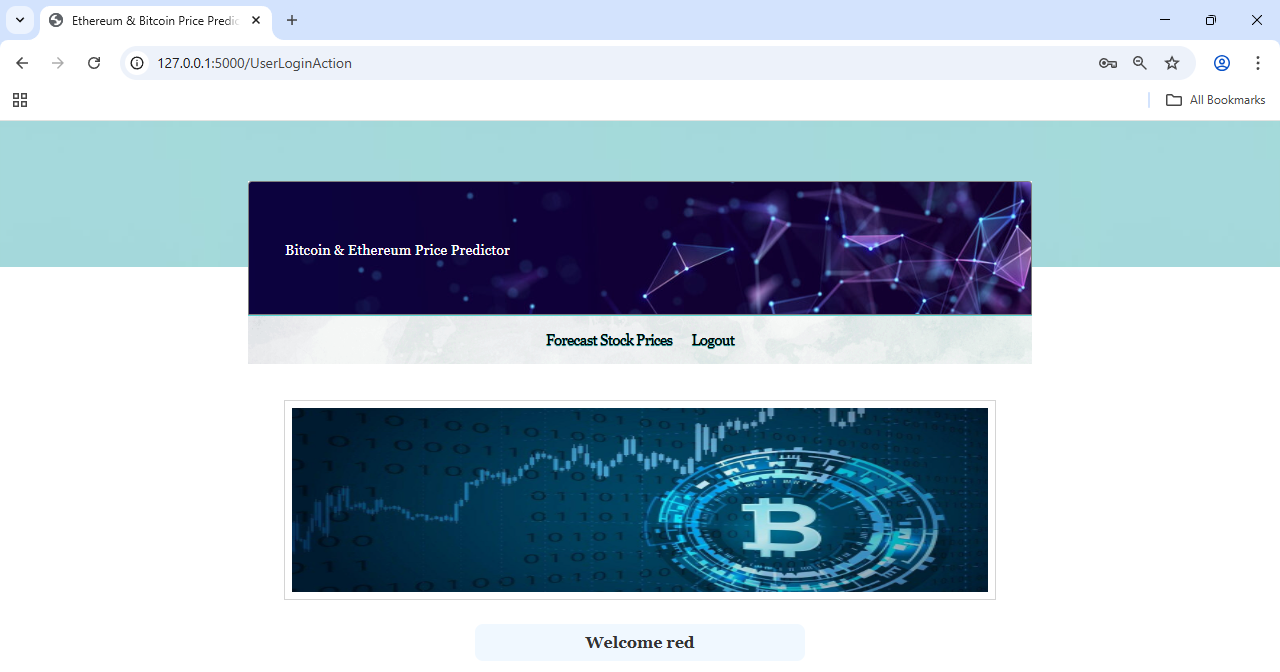


Fig. 6.6 Main page

Fig. 6.6 shows the Main page displayed after Login. It contains a welcome message with Forecast and Logout options in the navbar. The Logout returns the user to the Index page.

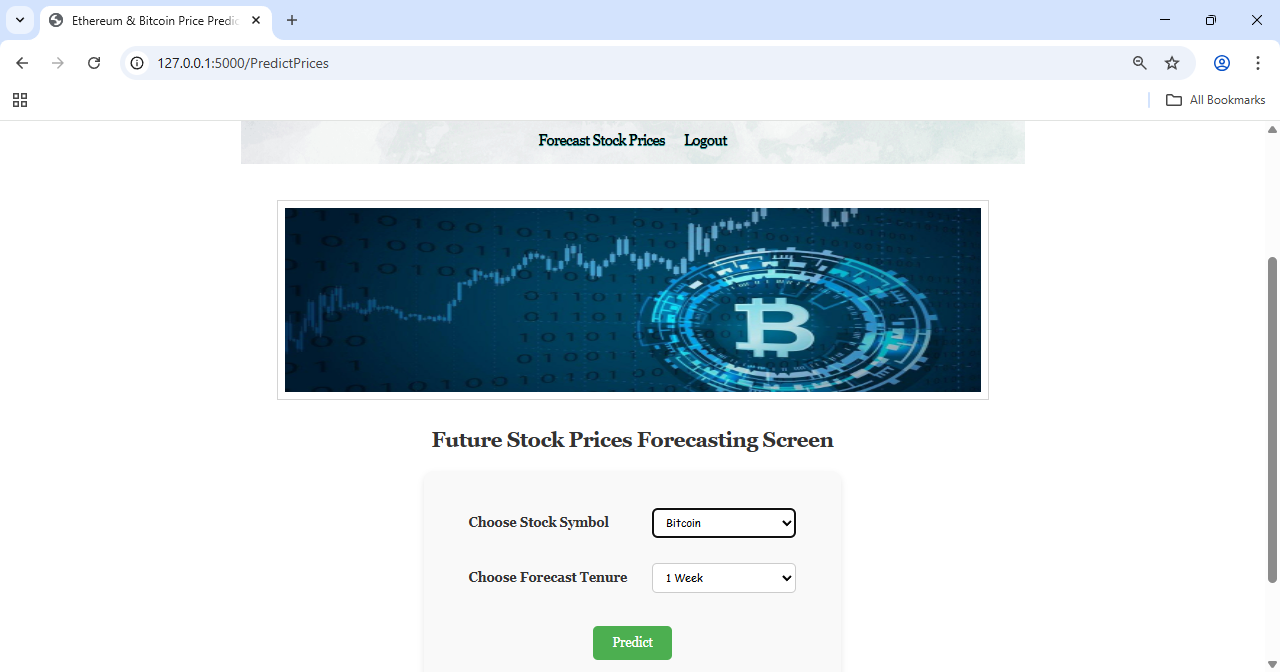


Fig. 6.7 Forecast Page

Fig. 6.7 shows the page displayed upon clicking Forecast Stock Prices. It allows us to select the type of coin and duration of prediction. The prediction results are shown upon clicking the predict button.

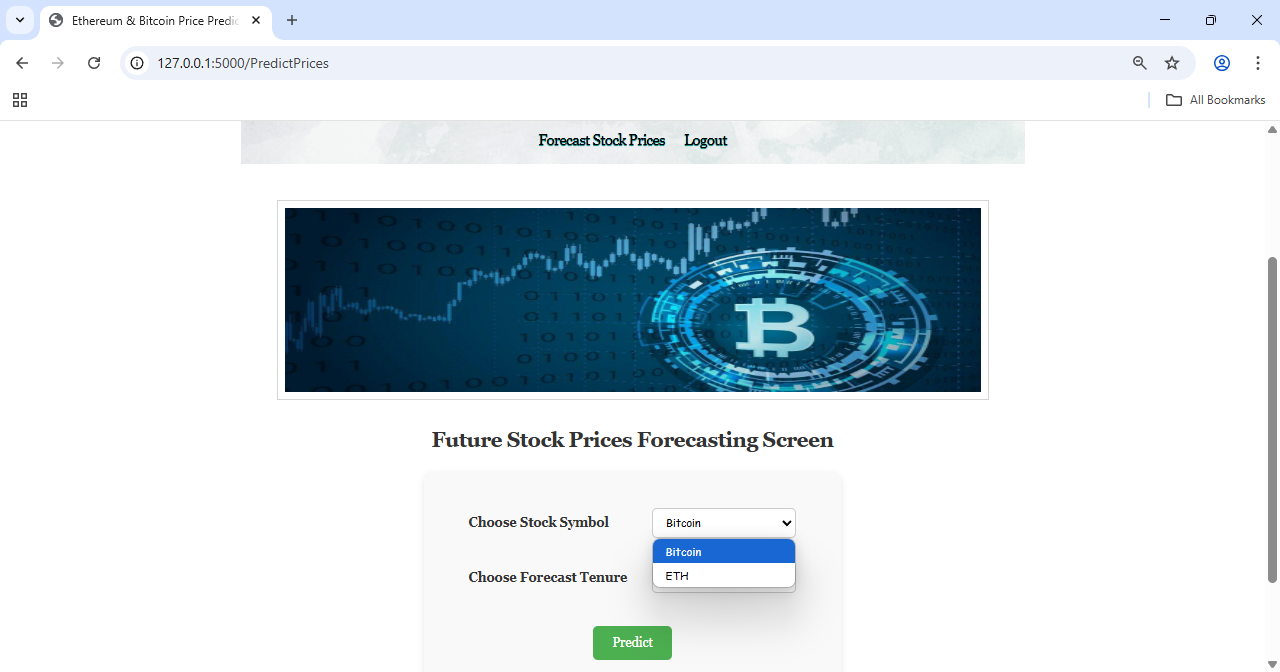


Fig. 6.8 Coin Select

Fig. 6.8 shows the options for selecting the type of coin.

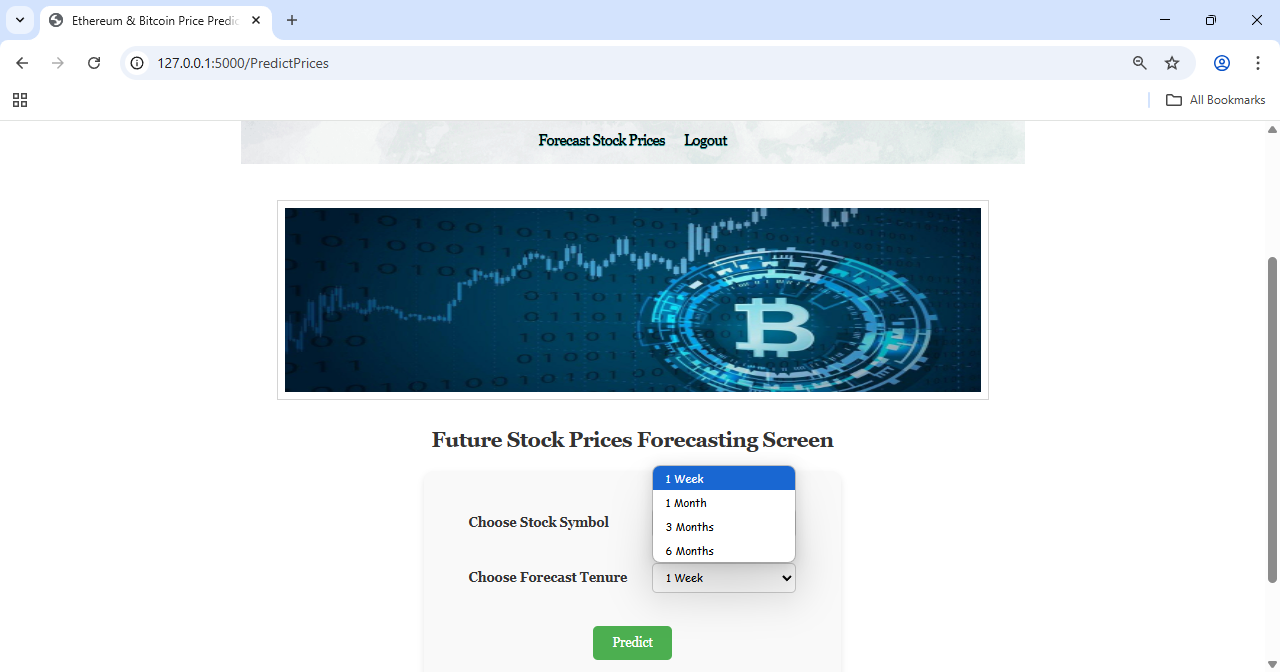


Fig. 6.9 Forecast Period Select

Fig. 6.9 shows the options for duration of forecasting the price of selected coin.

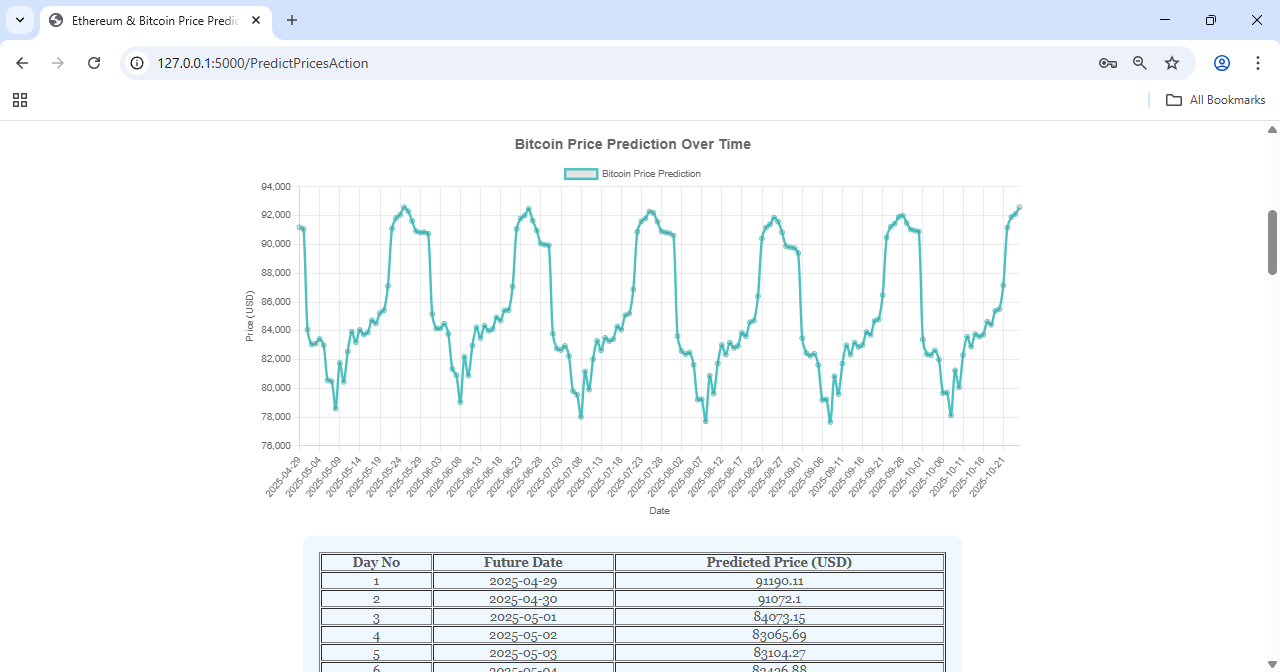
****

Fig. 6.10 Bitcoin 6-month forecast

Fig. 6.10 shows the results for predicting prices of bitcoin with 6 months forecast period. The other forecast periods work similarly but result in lesser records in table and lesser plotted points on graph.

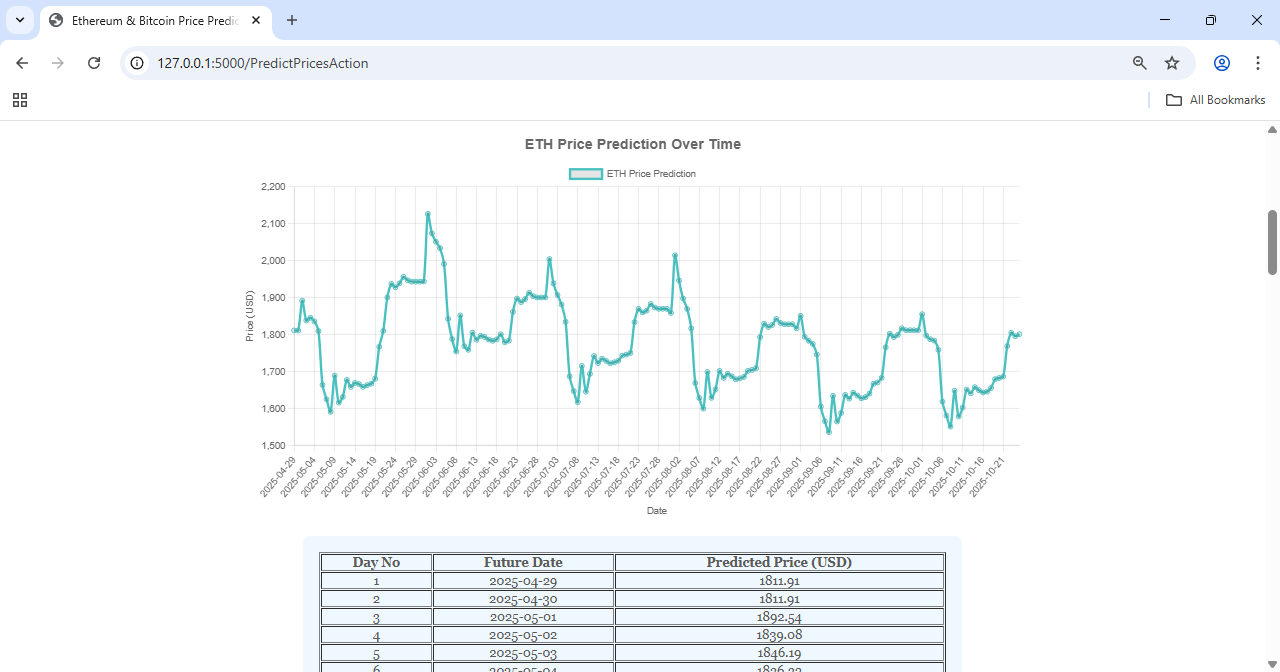
****

Fig. 6.11 Ethereum 6-month forecast

Fig. 6.11 shows the results for predicting prices of Ethereum with 6 months forecast period. The other forecast periods work similarly but result in lesser records in table and lesser plotted points on graph.

**7. CONCLUSION AND FUTURE ENHANCEMENTS**

**7.1 CONCLUSION**

The project, effectively demonstrates the application of advanced predictive techniques to tackle the complexities of cryptocurrency price forecasting. By leveraging a Voting Regressor model, the system combines the strengths of multiple algorithms to deliver accurate and reliable price predictions. The use of real-time data integration ensures that the predictions remain relevant and adaptable to the highly volatile nature of cryptocurrency markets.

The Voting Regressor model was chosen for its ability to aggregate predictions from various algorithms, thereby enhancing accuracy and minimizing biases or errors inherent in individual models. This approach allowed the system to overcome many limitations of traditional forecasting methods, including their inability to handle the rapid and unexpected changes characteristic of cryptocurrency markets.

**7.2 FUTURE ENHANCEMENT**

While the current system effectively leverages the Voting Regressor model for accurate cryptocurrency price forecasting, there is significant potential for further development. One major enhancement could involve integrating sentiment analysis from sources like social media, news platforms, and market reports could enable the system to capture the emotional and psychological factors that heavily influence cryptocurrency markets while also predicting future expected sentiments that are likely to arise to ensure the system can make predictions while including the future expected sentiments.

Another area of improvement lies in including features for risk assessment and portfolio optimization based on predicted price trends. Furthermore, expanding the model to simultaneously predict multiple cryptocurrencies and analyze their interdependencies could provide a more comprehensive view for traders and investors.

**REFERENCES**

1. M. Rafi, Q. A. K. Mirza, M. I. Sohail, M. Aliasghar, A. Aziz and S. Hameed, "Enhancing Cryptocurrency Price Forecasting Accuracy: A Feature Selection and Weighting Approach With Bi-Directional LSTM and Trend-Preserving Model Bias Correction," in IEEE Access, vol. 11, pp. 65700-65710, 2023, doi: 10.1109/ACCESS.2023.3287888.
2. H. Shamshad, F. Ullah, A. Ullah, V. R. Kebande, S. Ullah and A. Al-Dhaqm, "Forecasting and Trading of the Stable Cryptocurrencies With Machine Learning and Deep Learning Algorithms for Market Conditions," in IEEE Access, vol. 11, pp. 122205-122220, 2023, doi: 10.1109/ACCESS.2023.3327440.
3. G. Kim, D. -H. Shin, J. G. Choi and S. Lim, "A Deep Learning-Based Cryptocurrency Price Prediction Model That Uses On-Chain Data," in IEEE Access, vol. 10, pp. 56232-56248, 2022, doi: 10.1109/ACCESS.2022.3177888.
4. Z. Shahbazi and Y. -C. Byun, "Improving the Cryptocurrency Price Prediction Performance Based on Reinforcement Learning," in IEEE Access, vol. 9, pp. 162651-162659, 2021, doi: 10.1109/ACCESS.2021.3133937.
5. S. Tanwar, N. P. Patel, S. N. Patel, J. R. Patel, G. Sharma and I. E. Davidson, "Deep Learning-Based Cryptocurrency Price Prediction Scheme With Inter-Dependent Relations," in IEEE Access, vol. 9, pp. 138633-138646, 2021, doi: 10.1109/ACCESS.2021.3117848.
6. P. Jay, V. Kalariya, P. Parmar, S. Tanwar, N. Kumar and M. Alazab, "Stochastic Neural Networks for Cryptocurrency Price Prediction," in IEEE Access, vol. 8, pp. 82804-82818, 2020, doi: 10.1109/ACCESS.2020.2990659.
7. Yahoo Finance — <https://finance.yahoo.com/> (Data source used internally by yfinance.)