ANALYZING CRYPTO CURRENCY PRICES THROUGH THE LENS OF ARTIFICIAL INTELLIGENCE

A "Project Stage II" Report submitted to JNTU Hyderabad in partial fulfillment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

In

CSE (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

Submitted by

ALLA PRAKASH SAI	**20S11A6625**
BOMMAKANTI JASHWANTH	**20S11A6614**
REPALLE NANDINI	**20S11A6620**
PARUVELLI SOUMITH	**20S11A6639**

Under the Guidance of

Mrs. B. ANUSHA

B. Tech, M. Tech.

Assistant Professor of CSE(AI&ML)

DEPARTMENT OF CSE (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

MALLA REDDY INSTITUTE OF TECHNOLOGY & SCIENCE

(Approved by AICTE New Delhi and Affiliated to JNTUH)

(Accredited by NBA & NAAC with "A" Grade)

An ISO 9001: 2015 Certified Institution

Maisammaguda, Medchal (M), Hyderabad-500100, T. S.

MARCH 2024

DEPARTMENT OF CSE (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

MALLA REDDY INSTITUTE OF TECHNOLOGY & SCIENCE

(Approved by AICTE New Delhi and Affiliated to JNTUH)
(Accredited by NBA & NAAC with “A” Grade)
An ISO 9001: 2015 Certified Institution
Maisammaguda, Medchal (M), Hyderabad-500100, T. S.

MARCH 2024

CERTIFICATE

This is to certify that the Project Stage II entitled **“ANALYZING CRYPTO CURRENCY PRICES THROUGH THE LENS OF ARTIFICIAL INTELLIGENCE”** has been submitted by **ALLA PRAKASH SAI (20S11A6625), BOMMAKANTI JASHWANTH (20S11A6614), REPALLE NANDINI (20S11A6620),** and **PARUVELLI SOUMITH (20S11A6639)** in partial fulfillment of the requirements for the award of **BACHELOR OF TECHNOLOGY** in **CSE(AI&ML)**. This record of bonafide work carried out by them under my guidance and supervision. **The result embodied in this Project Stage II report has not been submitted to any other University or Institute for the award of any degree.**

Mrs. B. ANUSHA
Assistant Professor
Project Guide

DR. A. NAGARAJU
Head of the Department
CSE(AI&ML)

External Examiner

ACKNOWLEDGEMENT

The Project Stage II work carried out by our team in the Department of CSE (Artificial intelligence and Machine Learning), Malla Reddy Institute of Technology and Science, Hyderabad. ***This work is original and has not been submitted in part or full for any degree or diploma of any other university.***

We wish to acknowledge our sincere thanks to our project guide **Mrs. B. Anusha**, Assistant Professor for formulation of the problem, analysis, guidance and her continuous supervision during the course of work.

We acknowledge our sincere thanks to **Dr. Vaka Murali Mohan,** Principal and **Dr. A. Nagaraju**, Head of the Department and Coordinator, faculty members of CSE(AI&ML) Department for their kind cooperation in making this Project Stage II work a success.

We extend our gratitude to **Sri. Ch. Malla Reddy**, Founder Chairman MRGI and **Sri. Ch. Mahender Reddy,** Secretary MRGI, **Dr.Ch. Bhadra Reddy**, President MRGI, **Sri. Ch. Shalini Reddy**, Director MRGI, **Sri. P. Praveen Reddy**, Director MRGI, for their kind cooperation in providing the infrastructure for completion of our Project Stage II.

We acknowledge our special thanks to the entire teaching faculty and non-teaching staff members of the CSE (Artificial Intelligence and Machine Learning) Department for their support in making this project work a success.

ALLA PRAKASH SAI	**20S11A6625**	______________
BOMMAKANTI JASHWANTH	**20S11A6614**	______________
REPALLE NANDINI	**20S11A6620**	______________
PARUVELLI SOUMITH	**20S11A6639**	______________

INDEX

ABSTRACT

Cryptocurrency is playing an increasingly important role in reshaping the financial system due to its growing popular appeal and merchant acceptance. While many people are making investments in Cryptocurrency, the dynamical features, uncertainty, the predictability of Cryptocurrency are still mostly unknown, which dramatically risk the investments. It is a matter to try to understand the factors that influence the value formation. In this study, we use advanced artificial intelligence frameworks of fully connected Artificial Neural Network (ANN) and Long Short-Term Memory (LSTM) Recurrent Neural Network to analyze the price dynamics of Bitcoin, Ethereum, and Ripple. We find that ANN tends to rely more on long-term history while LSTM tends to rely more on short-term dynamics, which indicate the efficiency of LSTM to utilize useful information hidden in historical memory is stronger than ANN. However, given enough historical information ANN can achieve a similar accuracy, compared with LSTM. This study provides a unique demonstration that Cryptocurrency market price is predictable. However, the explanation of the predictability could vary depending on the nature of the involved machine-learning model.

LIST OF FIGURES

CHAPTER-1: SYSTEM ANALYSIS

1.1 EXISTING SYSTEM

The existing system for Analyzing Crypto Currency Prices Through the Lens of Artificial Intelligence typically involves the application of machine learning algorithms to historical price data. These algorithms may include regression models, decision trees, support vector machines, or more advanced techniques such as neural networks. Data preprocessing steps, such as normalization and feature scaling, are often employed to prepare the data for analysis. The system trains the AI models on historical price data and evaluates their performance using metrics such as accuracy, precision, and recall. While AI-based systems offer advantages such as automation and scalability, they also have limitations. These include the need for extensive computational resources, the risk of overfitting to historical data, and challenges in handling the inherent volatility and unpredictability of cryptocurrency markets. Despite these limitations, AI-based systems play a crucial role in providing insights into cryptocurrency price movements and informing investment decisions in today's dynamic financial landscape. Existing systems for cryptocurrency price analysis using AI can be categorized into several types based on their methodologies, features, and applications. Here are some common types:

- ❖ **Regression Models:** These systems utilize regression analysis to model the relationship between cryptocurrency prices and various explanatory variables. They may employ linear regression, polynomial regression, or other regression techniques to make predictions based on historical price data.
- ❖ **Time Series Forecasting Models:** These systems focus on analysing cryptocurrency price time series data to make future price predictions. Techniques such as Autoregressive Integrated Moving Average (ARIMA), Exponential Smoothing (ETS), or Prophet are commonly used for time series forecasting.
- ❖ **Machine Learning Algorithms:** These systems leverage a wide range of machine learning algorithms, including decision trees, random forests, support vector machines (SVM), k-nearest neighbours (KNN), and ensemble methods like gradient boosting and bagging. These algorithms can capture complex patterns and relationships in cryptocurrency price data.
- ❖ **Deep Learning Models:** These systems employ deep learning techniques such as artificial neural networks (ANNs) and recurrent neural networks (RNNs), including Long Short-Term Memory (LSTM) networks, to analyse cryptocurrency price dynamics. Deep learning

models are capable of learning intricate patterns in data and have shown promise in predicting cryptocurrency prices.

- **Sentiment Analysis Systems:** These systems analyse social media posts, news articles, and other sources of textual data to gauge market sentiment and its impact on cryptocurrency prices. Natural Language Processing (NLP) techniques are commonly used to extract sentiment-related features and sentiment scores from text data.
- **Algorithmic Trading Systems:** These systems use AI and machine learning algorithms to automate trading strategies based on cryptocurrency price predictions. They may incorporate risk management techniques, portfolio optimization algorithms, and execution strategies to maximize profits and minimize risks in cryptocurrency trading.
- **Market Monitoring and Surveillance Systems:** These systems monitor cryptocurrency markets in real-time to detect anomalies, market manipulation, and fraudulent activities. They use AI techniques such as anomaly detection, pattern recognition, and clustering to identify suspicious trading behaviours and protect market integrity.
- **Portfolio Management Systems:** These systems help investors manage their cryptocurrency portfolios by providing insights into asset allocation, risk exposure, and performance evaluation. They may employ AI-driven optimization algorithms to construct diversified portfolios and rebalance them over time based on market conditions.

Each type of existing system has its strengths and weaknesses, and the choice of system depends on factors such as the specific use case, data availability, computational resources, and desired level of accuracy and reliability.

1.1.1 DISADVANTAGES:

While existing systems for Analyzing Crypto Currency Prices Through the Lens of Artificial Intelligence offer numerous benefits, they also come with several disadvantages. Here are some common drawbacks associated with these systems:

- **Overfitting:** AI models trained on historical price data may suffer from overfitting, where they perform well on the training data but fail to generalize to new, unseen data. This can lead to inaccurate predictions and unreliable insights when applied to real-world cryptocurrency markets.

- **Data Limitations:** Cryptocurrency markets are relatively young and lack extensive historical data compared to traditional financial markets. Limited data availability can hinder the performance of AI models, especially those that require large datasets for training, validation, and testing.

- **Volatility and Uncertainty:** Cryptocurrency markets are highly volatile and subject to rapid fluctuations in prices due to various factors such as market sentiment, regulatory changes, technological developments, and macroeconomic events. AI models may struggle to capture and predict such complex and dynamic price movements accurately.
- **Complexity and Interpretability:** Deep learning models, such as artificial neural networks and recurrent neural networks, are complex and black-box in nature, making it challenging to interpret their internal workings and decision-making processes. Lack of transparency and interpretability can undermine trust and confidence in the predictions generated by these models.
- **Computational Resources:** Training and running AI models for cryptocurrency price analysis require significant computational resources, including high-performance hardware such as GPUs and TPUs. Limited access to computational resources may pose challenges for researchers, traders, and developers working with AI-driven systems.

1.2 PROPOSED SYSTEM

The proposed system for Analysing Crypto Currency Prices Through the Lens of Artificial Intelligence integrates advanced artificial intelligence (AI) frameworks, including Fully Connected Artificial Neural Network (ANN) and Long Short-Term Memory (LSTM) Recurrent Neural Network, to address the limitations of existing systems. By leveraging ANN and LSTM models, the proposed system aims to capture complex patterns and relationships in cryptocurrency price data, providing more accurate predictions and insights into market dynamics. The system incorporates data preprocessing techniques to prepare historical price data for analysis and employs rigorous evaluation methods to assess the performance of the AI models. Additionally, the proposed system includes features such as model comparison, validation of predictive capabilities, and interpretation of results to enhance transparency and trustworthiness. By combining cutting-edge AI technologies with robust analytical methodologies, the proposed system offers a sophisticated approach to cryptocurrency price analysis, enabling investors, traders, and market participants to make informed decisions in the rapidly evolving cryptocurrency market landscape.

1.2.1 ADVANTAGES

The proposed system for Analyzing Crypto Currency Prices Through the Lens of Artificial Intelligence using advanced AI frameworks offers several advantages:

- ❖ **Enhanced Predictive Accuracy:** By leveraging advanced AI techniques such as Fully Connected Artificial Neural Network (ANN) and Long Short-Term Memory (LSTM) Recurrent Neural Network, the proposed system can capture complex patterns and relationships in crypto currency price data with greater accuracy compared to traditional methods.
- ❖ **Comprehensive Analysis:** The system can conduct a comprehensive analysis of cryptocurrency price dynamics, including both short-term fluctuations and long-term trends, providing a more holistic understanding of market behavior.
- ❖ **Efficient Data Utilization:** ANN and LSTM models are adept at utilizing historical data effectively, enabling the system to extract valuable insights from large volumes of cryptocurrency market data and make more informed predictions.
- ❖ **Dynamic Adaptability:** The system's ability to adapt to changing market conditions and incorporate new information in real-time allows it to provide timely and relevant insights to investors, traders, and other market participants.
- ❖ **Transparent Evaluation:** The proposed system includes rigorous evaluation methods, such as model comparison and validation of predictive capabilities, ensuring transparency and reliability in the analysis process.
- ❖ **Informative Interpretation:** The system provides interpretable results, allowing users to understand the underlying factors driving cryptocurrency price movements and make well-informed decisions based on actionable insights.
- ❖ **Risk Mitigation:** By accurately predicting cryptocurrency price dynamics, the system helps investors and traders mitigate risks and optimize their investment strategies, leading to better portfolio performance and risk-adjusted returns.

Overall, the proposed system offers a powerful and sophisticated approach to cryptocurrency price analysis, empowering users with actionable insights and enabling them to navigate the complexities of the cryptocurrency market with confidence.

1.3 INTRODUCTION

Cryptocurrency is the peer-to-peer digital monitory and payment system that exist online via a controlled algorithm. When a miner cracks an algorithm to record a block of transactions to public ledger named blockchain and the cryptocurrency is created when the block is added to the blockchain. It allows people to store and transfer through encryption protocol and distributed network. Mining is a necessary and competitive component of the cryptocurrency system. The miner with more computational power has a better chance of finding a new coin than that of less. Bitcoin is the first and one of the leading digital currencies (its market capitalization had more than $ 7 billion in 2014, and then it increased significantly to $ 29 billion in 2017) which was first introduced by Satoshi Nakamoto in 2008. Among many features of bitcoin, the most impressive one is decentralization that it can remove the involvement of traditional financial sectors and monetary authorities effectively due to its blockchain network features. In addition, the electronic payment system of Bitcoin is based on cryptographic proof rather than the trust between each other as its transaction history cannot be changed unless redoing all proof of work of all blockchain, which play a critical role of being a trust intermediary and this can be widely used in reality such as recording charitable contribution to avoid corruption. Moreover, bitcoin has introduced the controllable anonymity scheme, and this enhances users' safety and anonymity by using this technology, for instance, we can take advantage of this property of blockchain to make identification cards, and it not only can protect our privacy but verify our identity. Nowadays, investing in cryptocurrencies, like Bitcoin, is one of the efficient ways of earning money. For example, the rate of Bitcoin significant rises in 2017, from a relatively low point 963 USD on January 1ST 2017, to its peak 19186 USD on December 17th 2017, and it closed with 9475 USD at the end of the year. Consequently, the rate of return of bitcoin investment for 2017 was over 880%, which is an impressive and surprising scenery for most investors.

While an increasing number of people are making investments in Cryptocurrency, the majority of investors cannot get such profit for being inconsiderable to cryptocurrencies' dynamics and the critical factors that influence the trends of bitcoins. Therefore, raising people's awareness of vital factors can help us to be wise investors. Although market prediction is demanding for its complex nature the dynamics are predictable and understandable to some degree. For example, when there is a shortage of the bitcoin, its price will be increased by their sellers as investors who regard bitcoin as a profitable investment opportunity will have a strong desire to pay for bitcoin. Furthermore, the price of bitcoin may be easily influenced by some influential external factors such as political factors. Although existing efforts on Cryptocurrency analysis and prediction is limited, a few studies have been aiming to understand the Cryptocurrency time series and build statistical models to reproduce and predict price dynamics. For example, Madan et al. collected bitcoins price with the time interval of 0.5, 1and 2 hours, and combined it with the blockchain network, the underlying technology of bitcoin. Their predictive model leveraging random forests and binomial logistic regression classifiers, and the precision of the model is around 55% in predicting bitcoin's price. Shah et al. used Bayesian regression and took advantages of high frequency (10-second) prices data of Bitcoin to improve investment strategy of bitcoin. Their models had also achieved great success. In an Multi-Layer Perceptron (MLP) based prediction model was presented to forecast the next day price of bitcoin by using two sets of input: the first type of inputs: the opening, minimum, maximum and closing price and the second set of inputs: Moving Average of both short (5,10,20 days) and long (100, 200 days) windows. During validation, their model was proved to be accurate at the 95% level. There has been many academic researches looking at exchange rate forecasting, for example, the monetary and portfolio balance models examined by Meese and Rogoff (1983, 1988). Significant efforts have been made to analyze and predict the trends of traditional financial markets especially the stock market however, predicting cryptocurrencies market prices is still at an early stage. Compared to these stock price prediction models, traditional time series methods are not very useful as cryptocurrencies are not precisely the same with stocks but can be deemed as a complementary good of existing currency system with sharp fluctuations features. Therefore, it is urgently needed to understand the dynamics of cryptocurrencies better and establish a suitable predictive modelling framework. In this study, we hypothesize that time series of cryptocurrencies exhibits a clear internal memory, which could be used to help the memory-based time series model to works more appropriately if the length of internal memory could be quantified. We aim to use two artificial intelligence modelling frameworks to understand and predict the most popular cryptocurrencies price dynamics, including Bitcoin, Ethereum, and Ripple.

CHAPTER 2: LITERATURE SURVEY

1) Using the Bitcoin Transaction Graph to Predict the Price of Bitcoin

AUTHORS: Greaves, A., & Au, B.

Bitcoin is the world’s leading cryptocurrency, allowing users to make transactions securely and anonymously over the Internet. In recent years, The Bitcoin the ecosystem has gained the attention of consumers, businesses, investors and speculators alike. While there has been significant research done to analyze the network topology of the Bitcoin network, limited research has been performed to analyze the network’s influence on overall Bitcoin price. In this paper, we investigate the predictive power of blockchain network-based features on the future price of Bitcoin. As a result of blockchain-network based feature engineering and machine learning optimization, we obtain up-down Bitcoin price movement classification accuracy of roughly 55%.

2) Cryptocurrency Value Formation: An Empirical Analysis Leading to A Cost of Production Model for Valuing Bitcoin

AUTHORS: Hayes, A. S.

This paper aims to identify the likely source(s) of value that cryptocurrencies exhibit in the marketplace using cross sectional empirical data examining 66 of the most used such 'coins'. A regression model was estimated that points to three main drivers of cryptocurrency value: the difficulty in 'mining 'for coins; the rate of unit production; and the cryptographic algorithm employed. These amount to relative differences in the cost of production of one coin over another at the margin, holding all else equal. Bitcoin-denominated relative prices were used, avoiding much of the price volatility associated with the dollar exchange rate. The resulting regression model can be used to better understand the drivers of relative value observed in the emergent area of cryptocurrencies. Using the above analysis, a cost of production model is proposed for valuing bitcoin, where the primary input is electricity. This theoretical model produces useful results for both an individual producer, by setting breakeven points to start and stop production, and for the bitcoin exchange rate on a macro level. Bitcoin production seems to resemble a competitive commodity market; in theory miners will produce until their marginal costs equal their marginal product.

3) Economic Prediction Using Neural Networks: The Case of IBM Daily Stock Returns

AUTHORS: H. White

A report is presented of some results of an ongoing project using neural-network modeling and learning techniques to search for and decode nonlinear regularities in asset price movements. The author focuses on the case of IBM common stock daily returns. Having to deal with the salient features of economic data highlights the role to be played by statistical inference and requires modifications to standard learning techniques which may prove useful in other contexts.

4) Designing a neural network for forecasting financial and economic time series

AUTHORS: Kaastra and M. Boyd

Artificial neural networks are universal and highly flexible function approximators first used in the fields of cognitive science and engineering. In recent years, neural network applications in finance for such tasks as pattern recognition, classification, and time series forecasting have dramatically increased. However, the large number of parameters that must be selected to develop a neural network forecasting model have meant that the design process still involves much trial and error. The objective of this paper is to provide a practical introductory guide in the design of a neural network for forecasting economic time series data. An eight-step procedure to design a neural network forecasting model is explained including a discussion of tradeoffs in parameter selection, some common pitfalls, and points of disagreement among practitioners.

CHAPTER 3: SYSTEM DESIGN

3.1 SYSTEM ARCHITECTURE

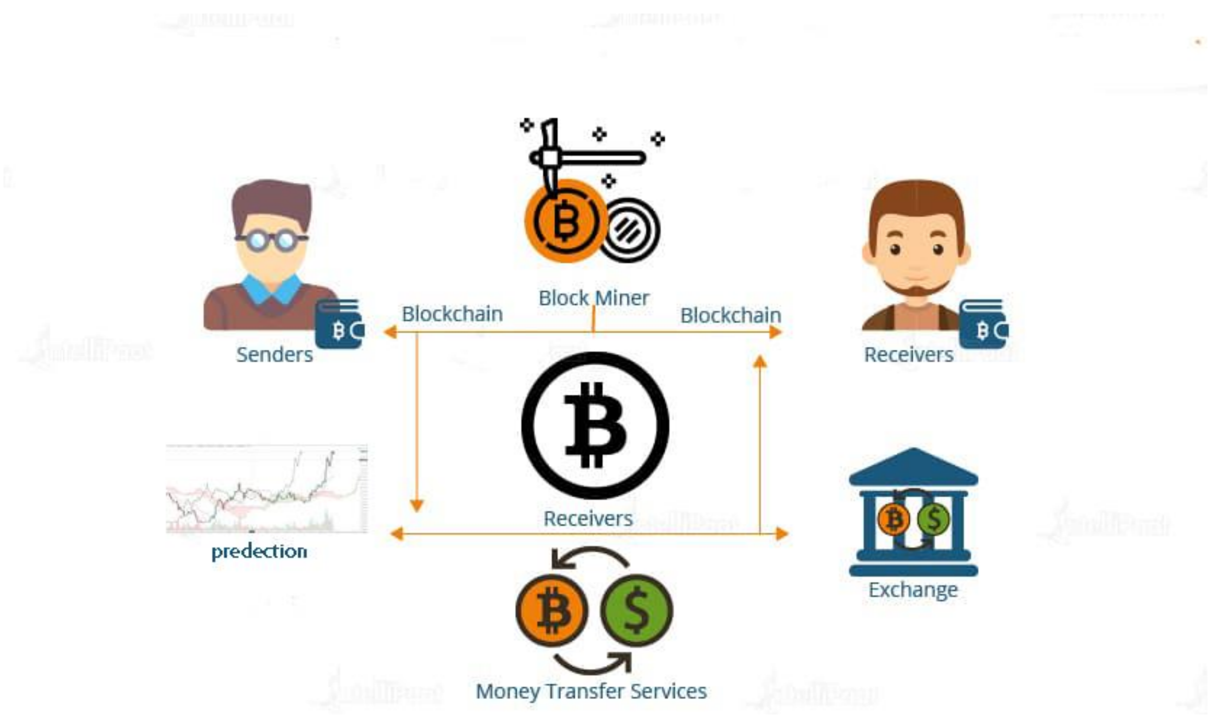


Fig 1: System Architecture

3.1.1 DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

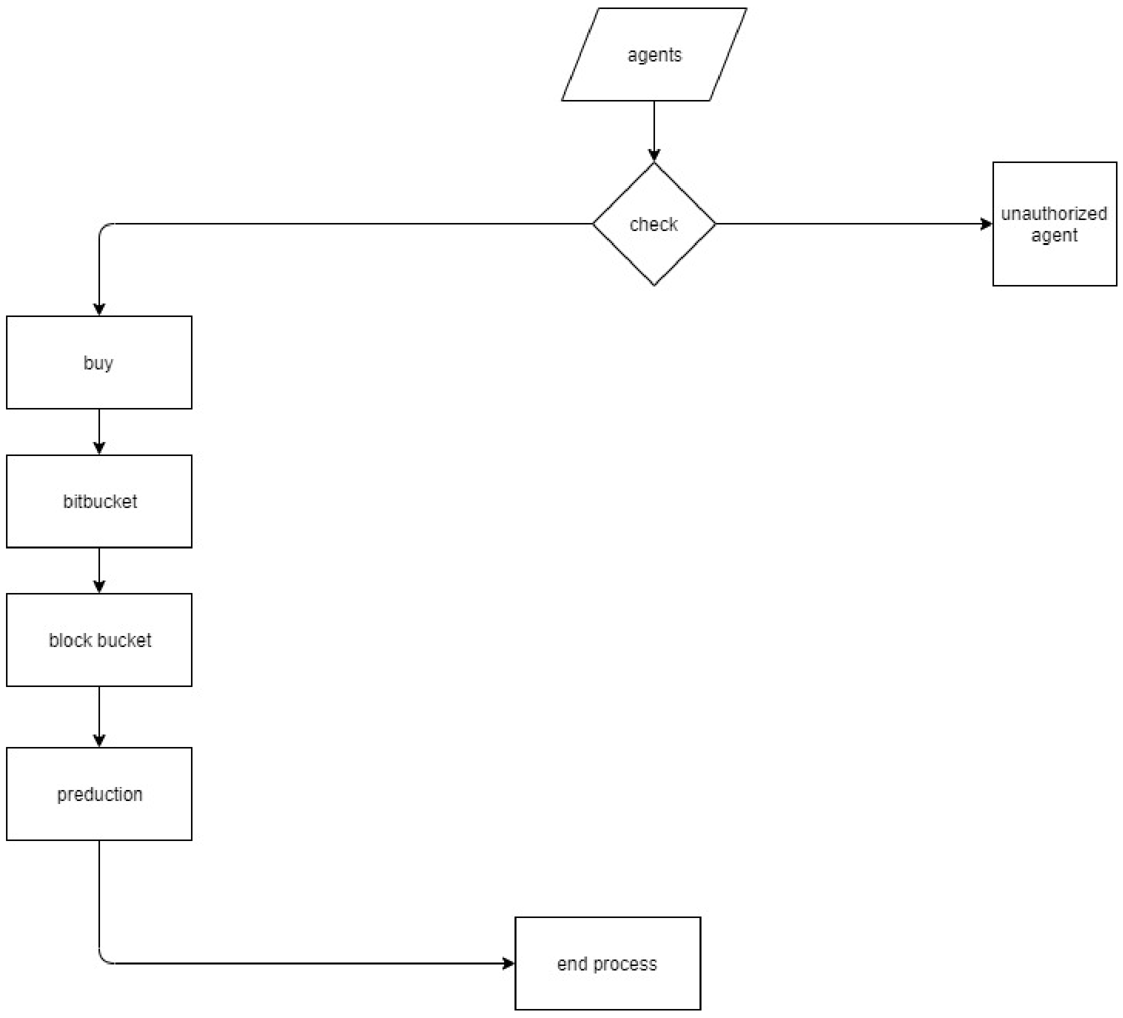


Fig 2: Data Flow Diagram

3.2 UML DIAGRAMS:

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

3.2.1 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

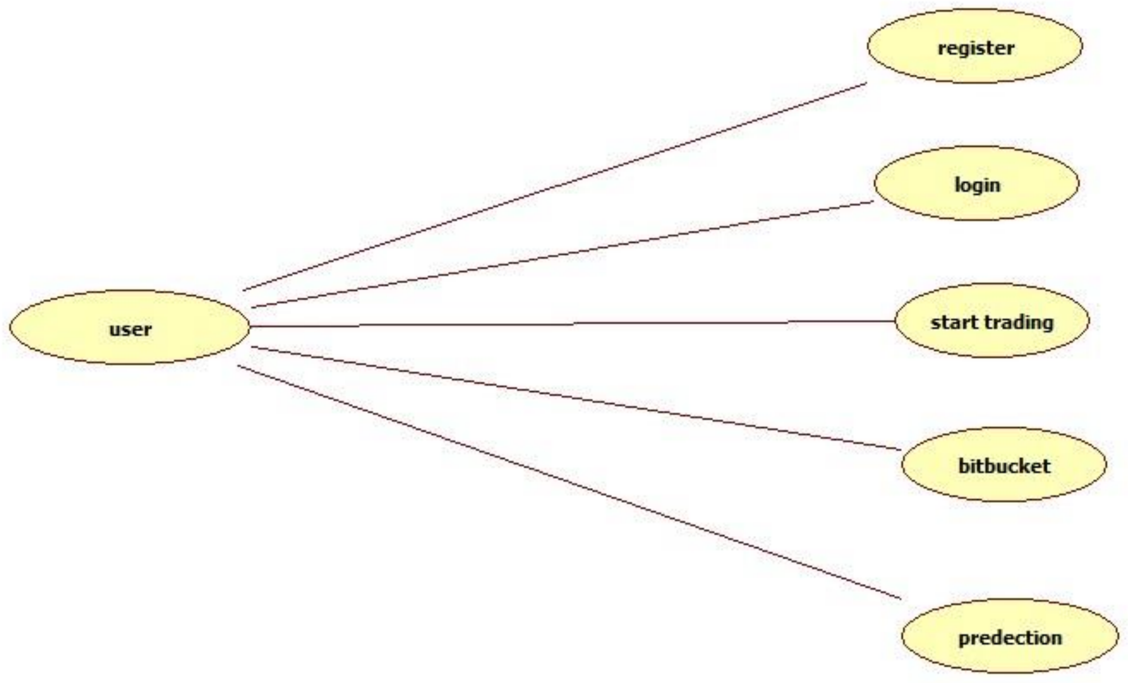


Fig 3: Use Case Diagram for User

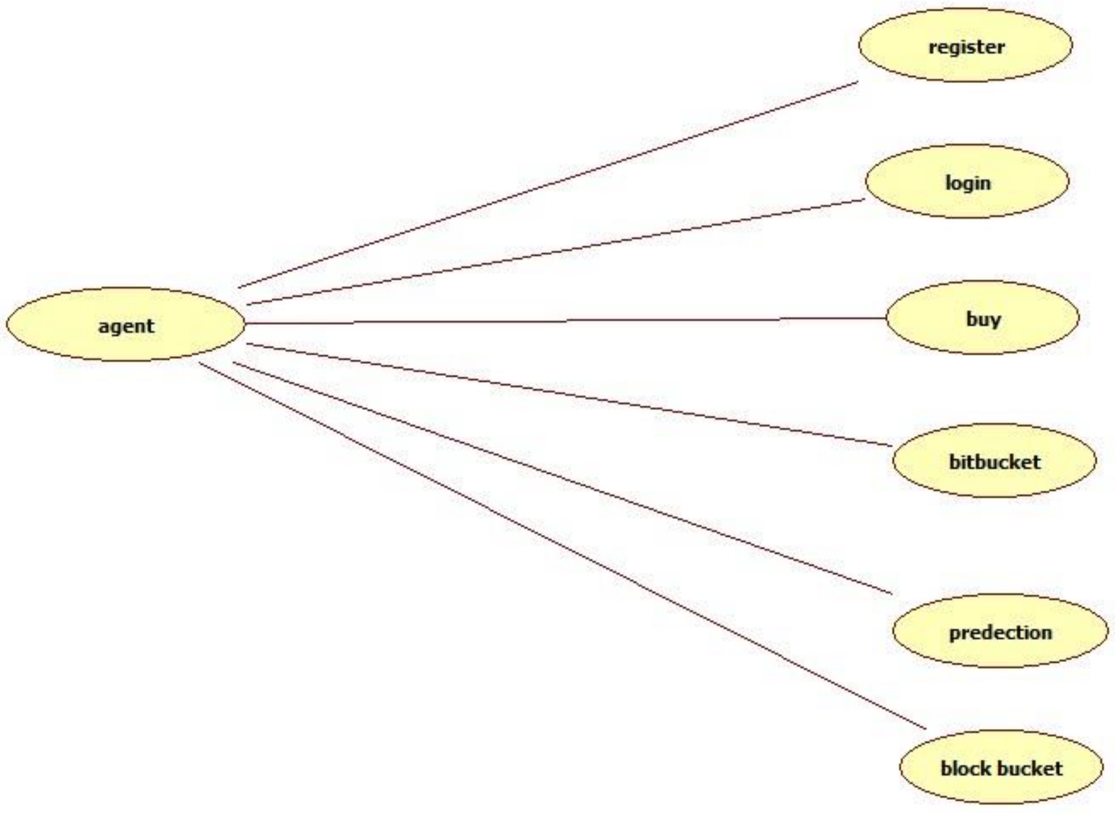


Fig 4: Use Case Diagram for Agent

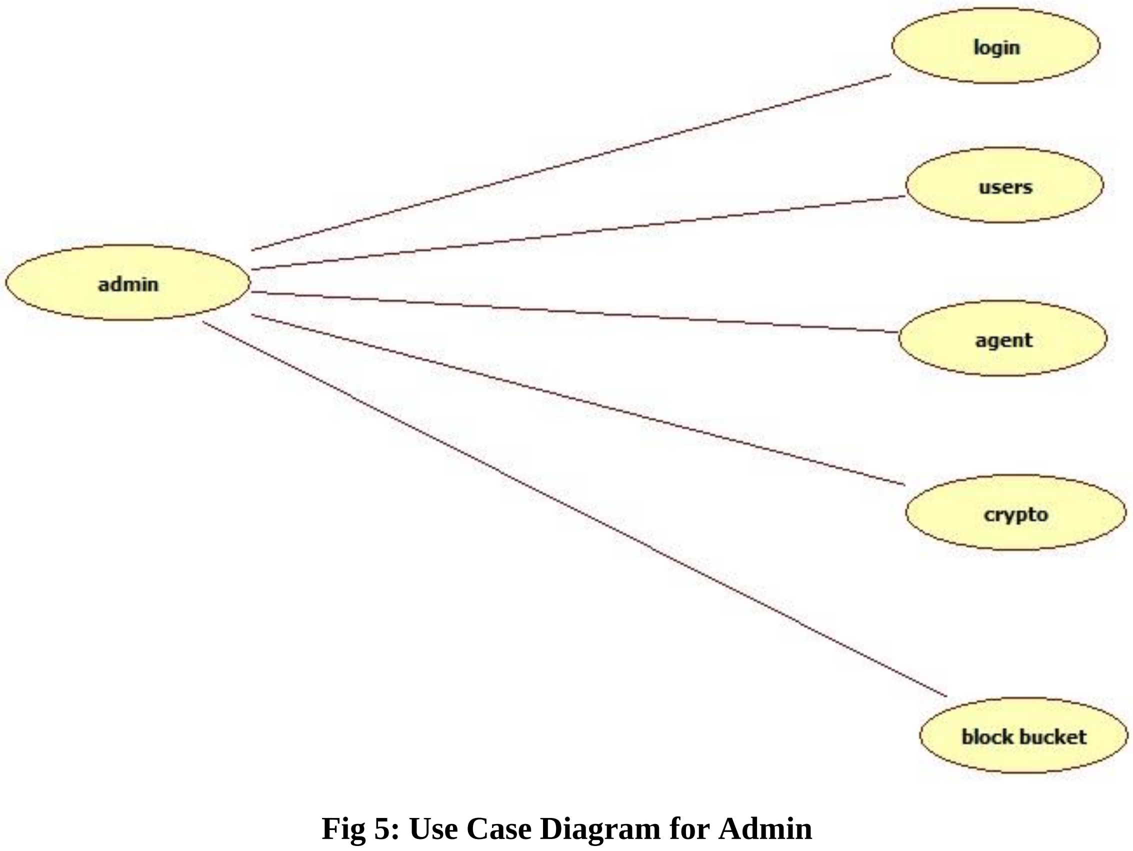


Fig 5: Use Case Diagram for Admin

3.2.2 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

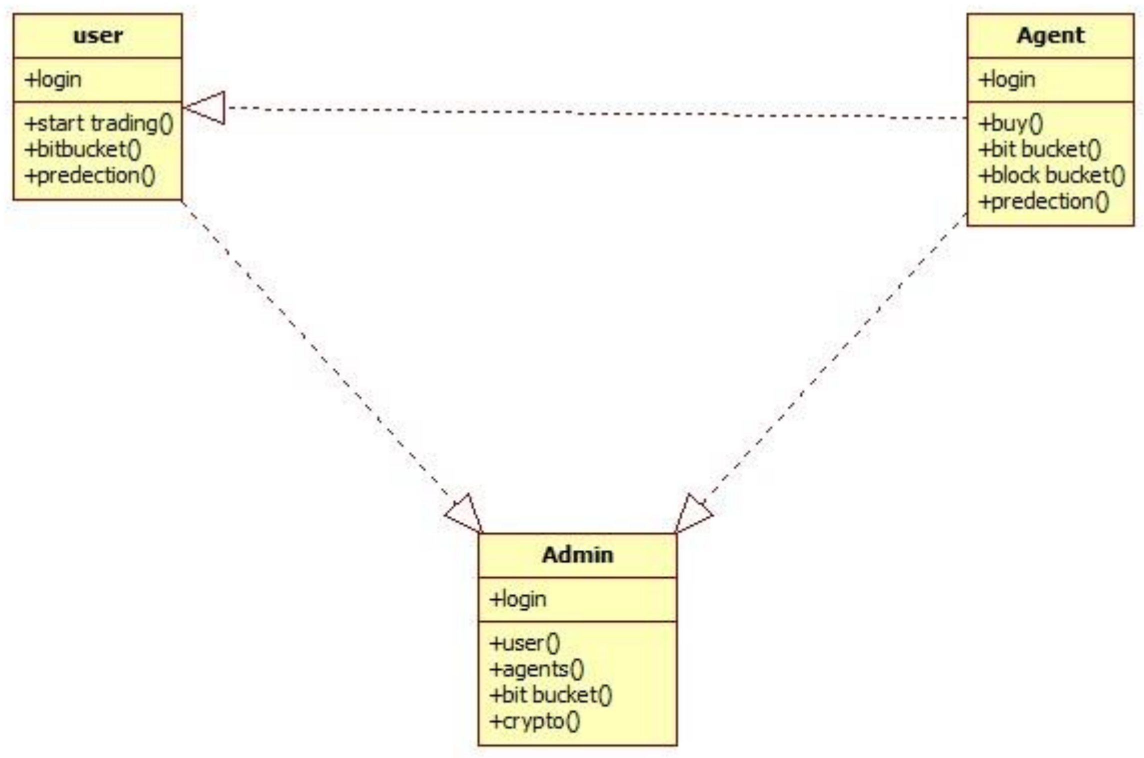


Fig 6: Class Diagram

3.2.3 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

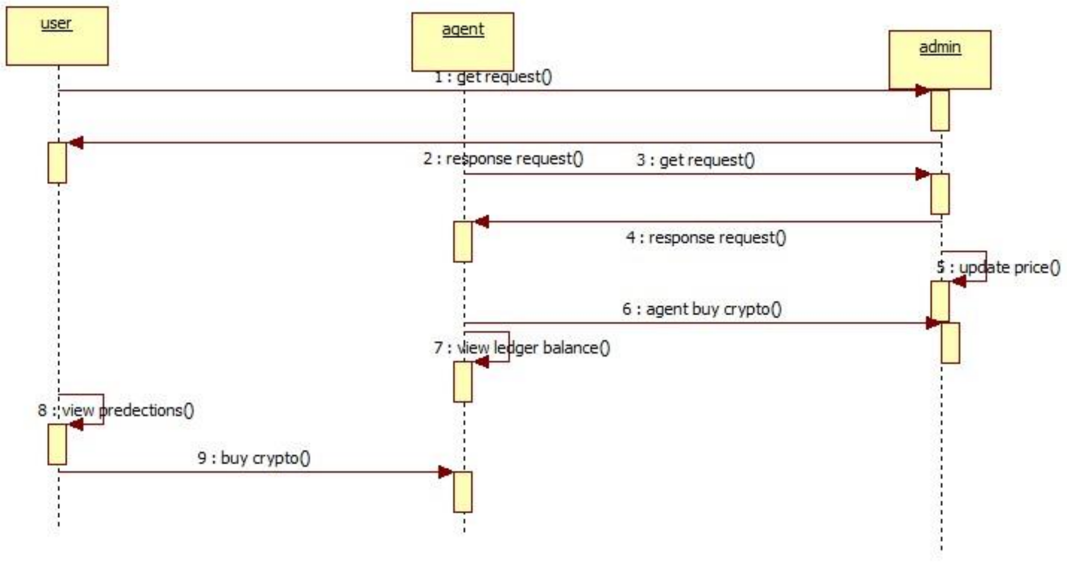


Fig 7: Sequence Diagram

3.2.4 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

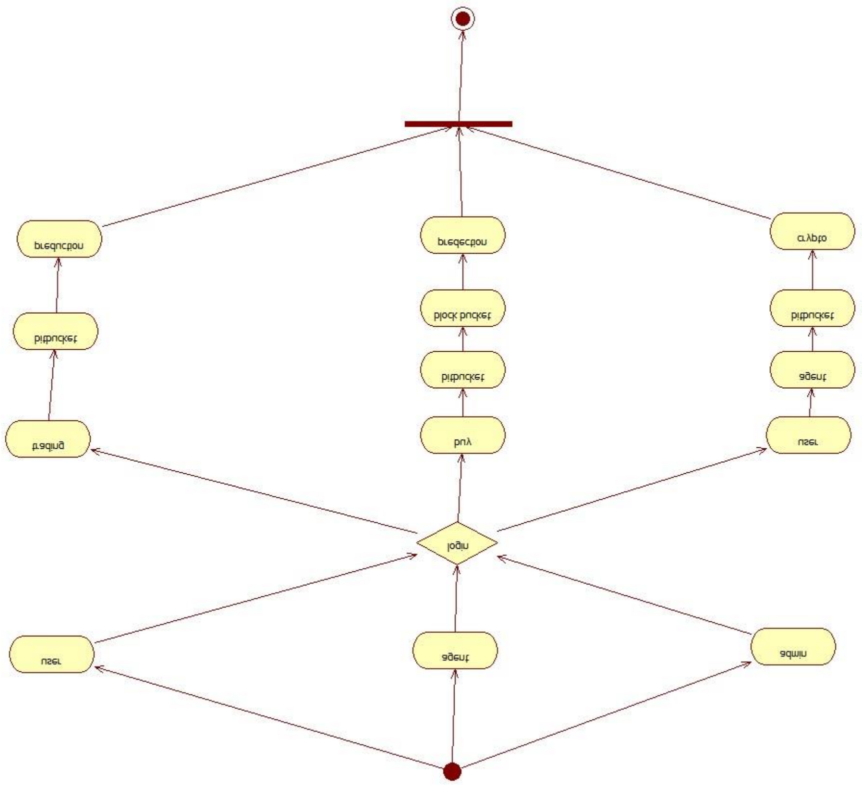


Fig 8: Activity Diagram

3.3 SYSTEM REQUIREMENTS

3.3.1 HARDWARE REQUIREMENTS:

❖ **System**	**:**	Pentium IV 2.4 GHz.
❖ **Hard Disk**	**:**	40 GB.
❖ **Floppy Drive**	**:**	1.44 Mb.
❖ **Monitor**	**:**	14’ Colour Monitor.
❖ **Mouse**	**:**	Optical Mouse.
❖ **Ram**	**:**	512 Mb.

3.3.2 SOFTWARE REQUIREMENTS:

❖ **Operating system**	**:**	Windows 7 Ultimate.
❖ **Coding Language**	**:**	Python.
❖ **Front-End**	**:**	Python.
❖ **Designing**	**:**	HTML, CSS, JAVASCRIPT.
❖ **Data Base**	**:**	MySQL.
❖ **Framework**	**:**	DJANGO

3.4 MODULES:

- User
- Agent
- Admin
- Artificial Intelligence

3.4.1 USER:

The electronic payment system of Bitcoin is based on cryptographic proof rather than the trust between each other as its transaction history cannot be changed unless redoing all proof of work of all blockchain, which play a critical role of being a trust intermediary and this can be widely used in reality such as recording charitable contribution to avoid corruption. Moreover, bitcoin has introduced the controllable anonymity scheme, and this enhances users’ safety and anonymity by using this technology, for instance, we can take advantage of this property of blockchain to make identification cards, and it not only can protect our privacy but verify our identity.

3.4.2 AGENT:

While an increasing number of people are making investments in Cryptocurrency, the majority of investors cannot get such profit for being inconsiderable to cryptocurrencies' dynamics and the critical factors that influence the trends of bitcoins. Therefore, raising people's awareness of vital factors can help us to be wise investors. Although market prediction is demanding for its complex nature, the dynamics are predictable and understandable to some degree. For example, when there is a shortage of the bitcoin, its price will be increased by their sellers as investors who regard bitcoin as a profitable investment opportunity will have a strong desire to pay for bitcoin. Furthermore, the price of bitcoin may be easily influenced by some influential external factors such as political factors.

3.4.3 ADMIN:

The aim of admin is to approve the users and agents. When a miner cracks an algorithm to record a block of transactions to public ledger named blockchain and the cryptocurrency is created when the block is added to the blockchain. It allows people to store and transfer through encryption protocol and distributed network. Mining is a necessary and competitive component of the cryptocurrency system. The miner with more computational power has a better chance of finding a new coin than that of less. Bitcoin is the first and one of the leading digital currencies (its market capitalization had more than $ 7 billion in 2014, and then it increased significantly to $ 29 billion in 2017) which was first introduced by Satoshi Nakamoto in 2008. Among many features of bitcoin, the most impressive one is decentralization that it can remove the involvement of traditional financial sectors and monetary authorities effectively due to its blockchain network features.

3.4.4 ARTIFICIAL INTELLIGENCE:

The application of advanced digital, smart technologies, robotic systems, new materials and design techniques, creation of large data processing systems, computer-aided learning and artificial intelligence (AI) are relevant for various branches of science and technology, including manned space programs. Some technology concepts and pilot systems based on the AI (3-D computer vision, automated systems for planning and evaluating the activities of cosmonauts, inquiry and communications system) were developed in the industry over several decades.

CHAPTER 4: INPUT AND OUTPUT DESIGN

4.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus, the objective of input design is to create an input layout that is easy to follow

4.2 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. `Create document, report, or other formats that contain information produced by the system. The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

CHAPTER 5: SYSTEM ENVIRONMENT

PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open-source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt −

```
$ python
Python 2.4.3 (#1, Nov 11 2010, 13:34:43)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Type the following text at the Python prompt and press the Enter −

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, Python!");. However, in Python version 2.4.3, this produces the following result −

```
Hello, Python!
```

Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file −

Live Demo

```
print "Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows −

```
$ python test.py
```

This produces the following result −

```
Hello, Python!
```

Let us try another way to execute a Python script. Here is the modified test.py file −

Live Demo

```
#!/usr/bin/python
print "Hello, Python!"
```

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows −

```
$ chmod +x test.py     # This is to make file executable
$./test.py
```

This produces the following result −

```
Hello, Python!
```

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers −

- Class names start with an uppercase letter. All other identifiers start with a lowercase letter.
- Starting an identifier with a single leading underscore indicates that the identifier is private.
- Starting an identifier with two leading underscores indicates a strongly private identifier.

- If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

Reserved Words

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and	exec	not
assert	finally	or
break	for	pass
class	from	print
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield

Lines and Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

```
if True:
   print "True"
else:
   print "False"
```

However, the following block generates an error −

```
if True:
print "Answer"
print "True"
else:
print "Answer"
print "False"
```

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks −

Note − Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces.

```
#!/usr/bin/python
import sys
try:
  # open file stream
  file = open(file_name, "w")
except IOError:
  print "There was an error writing to", file_name
  sys.exit()
print "Enter '", file_finish,
print "' When finished"
while file_text != file_finish:
  file_text = raw_input("Enter text: ")
  if file_text == file_finish:
    # close the file
    file.close
    break
  file.write(file_text)
  file.write("\n")
file.close()
file_name = raw_input("Enter filename: ")
if len(file_name) == 0:
  print "Next time please enter something"
  sys.exit()
try:
  file = open(file_name, "r")
except IOError:
  print "There was an error reading file"
  sys.exit()
file_text = file.read()
```

```
file.close()
print file_text
```

Multi-Line Statements

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example −

```
total = item_one + \
        item_two + \
        item_three
```

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example −

```
days = ['Monday', 'Tuesday', 'Wednesday',
        'Thursday', 'Friday']
```

Quotation in Python

Python accepts single ('), double (") and triple (''' or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal −

```
word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""
```

Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

Live Demo

```
#!/usr/bin/python
# First comment
print "Hello, Python!" # second comment
```

This produces the following result −

```
Hello, Python!
```

You can type a comment on the same line after a statement or expression −

```
name = "Madisetti" # This is again comment
```

You can comment multiple lines as follows −

```
# This is a comment.
# This is a comment, too.
# This is a comment, too.
# I said that already.
```

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comment:

```
'''
This is a multiline
comment.
'''
```

Using Blank Lines

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

In an interactive interpreter session, you must enter an empty physical line to terminate a multiline statement.

Waiting for the User

The following line of the program displays the prompt, the statement saying "Press the enter key to exit", and waits for the user to take action −

```
#!/usr/bin/python
raw_input("\n\nPress the enter key to exit.")
```

Here, "\n\n" is used to create two new lines before displaying the actual line. Once the user presses the key, the program ends. This is a nice trick to keep a console window open until the user is done with an application.

Multiple Statements on a Single Line

The semicolon (;) allows multiple statements on the single line given that neither statement starts a new code block. Here is a sample snip using the semicolon.

```
import sys; x = 'foo'; sys.stdout.write(x + '\n')
```

Multiple Statement Groups as Suites

A group of individual statements, which make a single code block are called suites in Python. Compound or complex statements, such as if, while, def, and class require a header line and a suite.

Header lines begin the statement (with the keyword) and terminate with a colon (:) and are followed by one or more lines which make up the suite. For example −

```
if expression :
   suite
elif expression :
   suite
else :
   suite
```

Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

```
$ python -h
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
Options and arguments (and corresponding environment variables):
-c cmd : program passed in as string (terminates option list)
-d     : debug output from parser (also PYTHONDEBUG=x)
-E     : ignore environment variables (such as PYTHONPATH)
-h     : print this help message and exit
```

You can also program your script in such a way that it should accept various options. Command Line Arguments is an advanced topic and should be studied a bit later once you have gone through rest of the Python concepts.

Python Lists

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"]
```

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example −

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5 );
tup3 = "a", "b", "c", "d";
```

The empty tuple is written as two parentheses containing nothing −

```
tup1 = ();
```

To write a tuple containing a single value you have to include a comma, even though there is only one value −

```
tup1 = (50,);
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example −

Live Demo

```
#!/usr/bin/python
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5, 6, 7 );
print "tup1[0]: ", tup1[0];
print "tup2[1:5]: ", tup2[1:5];
```

When the above code is executed, it produces the following result −

```
tup1[0]:  physics
tup2[1:5]:  [2, 3, 4, 5]
```

Updating Tuples

Accessing Values in Dictionary

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example −

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "dict['Name']: ", dict['Name']
print "dict['Age']: ", dict['Age']
```

When the above code is executed, it produces the following result −

```
dict['Name']:  Zara
dict['Age']:  7
```

If we attempt to access a data item with a key, which is not part of the dictionary, we get an error as follows −

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "dict['Alice']: ", dict['Alice']
```

When the above code is executed, it produces the following result −

```
dict['Alice']:
Traceback (most recent call last):
   File "test.py", line 4, in <module>
      print "dict['Alice']: ", dict['Alice'];
KeyError: 'Alice'
```

Updating Dictionary

You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example −

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
dict['Age'] = 8; # update existing entry
dict['School'] = "DPS School"; # Add new entry
print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

When the above code is executed, it produces the following result −

```
dict['Age']:  8
dict['School']:  DPS School
```

Delete Dictionary Elements

You can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the del statement. Following is a simple example −

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
del dict['Name']; # remove entry with key 'Name'
dict.clear();     # remove all entries in dict
del dict ;        # delete entire dictionary
print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

This produces the following result. Note that an exception is raised because after del dict dictionary does not exist anymor −

```
dict['Age']:
Traceback (most recent call last):
  File "test.py", line 8, in <module>
    print "dict['Age']: ", dict['Age'];
TypeError: 'type' object is unsubscriptable
```

Note − del() method is discussed in subsequent section.

Properties of Dictionary Keys

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys −

(a) More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins. For example −

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'}
print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result −

```
dict['Name']:  Manni
```

(b) Keys must be immutable. Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed. Following is a simple example −

Live Demo

```
#!/usr/bin/python
dict = {['Name']: 'Zara', 'Age': 7}
```

```
print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result −

```
Traceback (most recent call last):
   File "test.py", line 3, in <module>
      dict = {['Name']: 'Zara', 'Age': 7};
TypeError: unhashable type: 'list'
```

Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates −

 Live Demo

```
#!/usr/bin/python
tup1 = (12, 34.56);
tup2 = ('abc', 'xyz');
# Following action is not valid for tuples
# tup1[0] = 100;
# So let's create a new tuple as follows
tup3 = tup1 + tup2;
print tup3;
```

When the above code is executed, it produces the following result −

```
(12, 34.56, 'abc', 'xyz')
```

Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the del statement. For example −

 Live Demo

```
#!/usr/bin/python
tup = ('physics', 'chemistry', 1997, 2000);
print tup;
del tup;
print "After deleting tup : ";
print tup;
```

This produces the following result. Note an exception raised, this is because after del tup tuple does not exist anymore −

```
('physics', 'chemistry', 1997, 2000)
```

```
After deleting tup:
Traceback (most recent call last):
  File "test.py", line 9, in <module>
    print tup;
NameError: name 'tup' is not defined
```

DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.

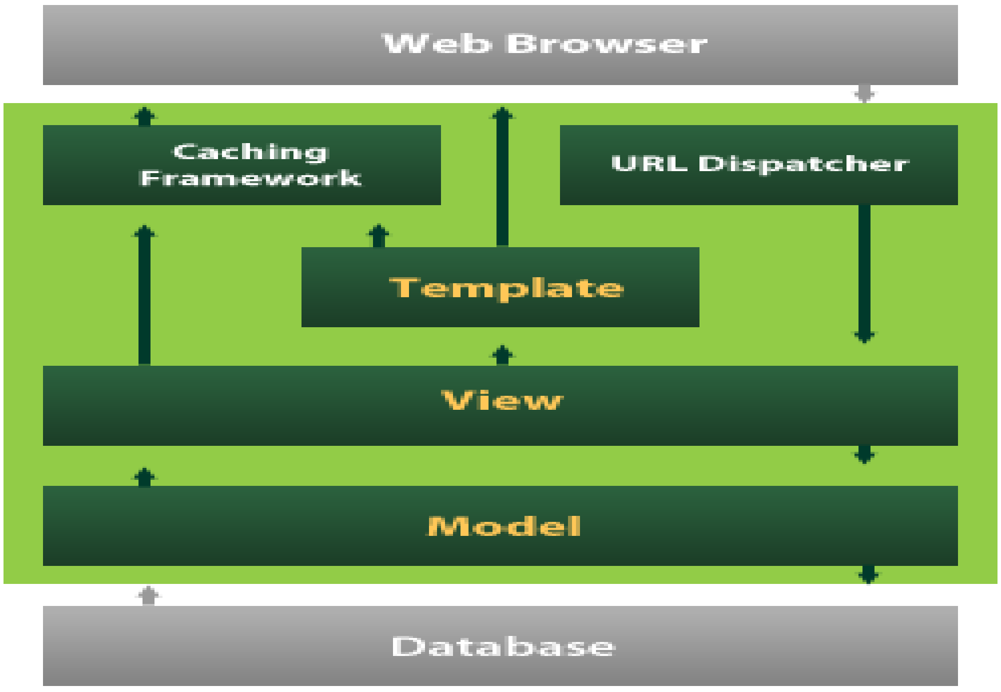


Fig 9: Django Framework

Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models

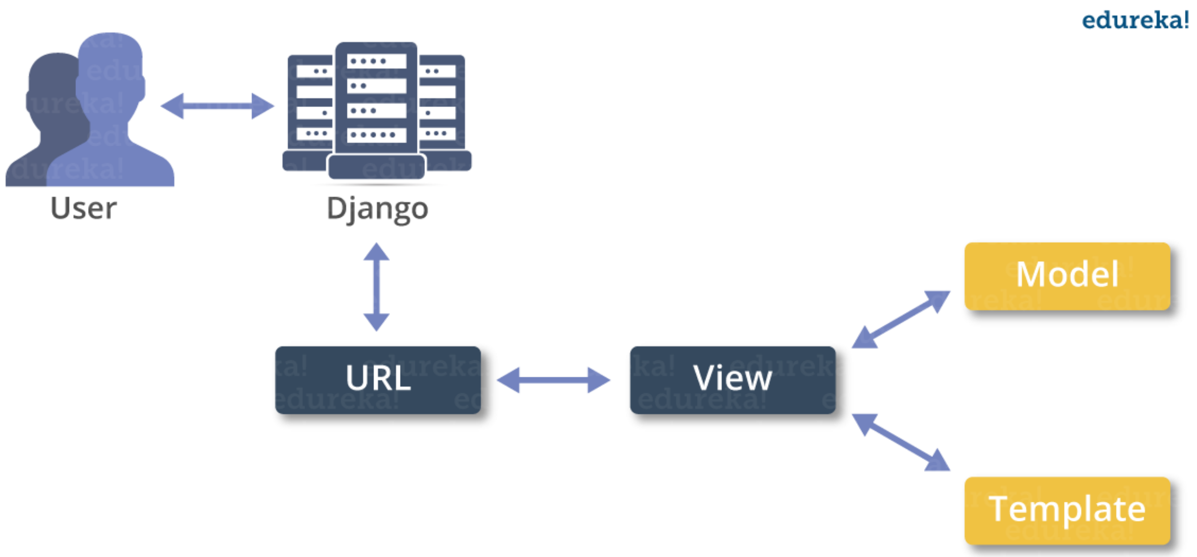


Fig 10: Django Workflow

Create a Project

Whether you are on Windows or Linux, just get a terminal or a cmd prompt and navigate to the place you want your project to be created, then use this code –

```
$ django-admin startproject myproject
```

This will create a "myproject" folder with the following structure –

```
myproject/
   manage.py
   myproject/
      __init__.py
      settings.py
      urls.py
      wsgi.py
```

The Project Structure

The “myproject” folder is just your project container, it actually contains two elements –

manage.py – This file is kind of your project local django-admin for interacting with your project via command line (start the development server, sync db...). To get a full list of command accessible via manage.py you can use the code –

```
$ python manage.py help
```

The “myproject” subfolder – This folder is the actual python package of your project. It contains four files –

__init__.py – Just for python, treat this folder as package.

settings.py – As the name indicates, your project settings.

urls.py − All links of your project and the function to call. A kind of ToC of your project.
wsgi.py − If you need to deploy your project over WSGI.
Setting Up Your Project
Your project is set up in the subfolder myproject/settings.py. Following are some important options you might need to set −

```
DEBUG = True
```

This option lets you set if your project is in debug mode or not. Debug mode lets you get more information about your project's error. Never set it to 'True' for a live project. However, this has to be set to 'True' if you want the Django light server to serve static files. Do it only in the development mode.

```
DATABASES = {
   'default': {
      'ENGINE': 'django.db.backends.sqlite3',
      'NAME': 'database.sql',
      'USER': '',
      'PASSWORD': '',
      'HOST': '',
      'PORT': '',
   }
}
```

Database is set in the 'Database' dictionary. The example above is for SQLite engine. As stated earlier, Django also supports −
MySQL (django.db.backends.mysql)
PostGreSQL (django.db.backends.postgresql_psycopg2)
Oracle (django.db.backends.oracle) and NoSQL DB
MongoDB (django_mongodb_engine)
Before setting any new engine, make sure you have the correct db driver installed.
You can also set others options like: TIME_ZONE, LANGUAGE_CODE, TEMPLATE…
Now that your project is created and configured make sure it's working −

```
$ python manage.py runserver
```

You will get something like the following on running the above code −

```
Validating models...
0 errors found
```

```
September 03, 2015 - 11:41:50
Django version 1.6.11, using settings 'myproject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

A project is a sum of many applications. Every application has an objective and can be reused into another project, like the contact form on a website can be an application, and can be reused for others. See it as a module of your project.

Create an Application

We assume you are in your project folder. In our main “myproject” folder, the same folder then manage.py −

```
$ python manage.py startapp myapp
```

You just created myapp application and like project, Django create a “myapp” folder with the application structure −

```
myapp/
   __init__.py
   admin.py
   models.py
   tests.py
   views.py
```

__init__.py − Just to make sure python handles this folder as a package.

admin.py − This file helps you make the app modifiable in the admin interface.

models.py − This is where all the application models are stored.

tests.py − This is where your unit tests are.

views.py − This is where your application views are.

Get the Project to Know About Your Application

At this stage we have our "myapp" application, now we need to register it with our Django project "myproject". To do so, update INSTALLED_APPS tuple in the settings.py file of your project (add your app name) −

```
INSTALLED_APPS = (
   'django.contrib.admin',
   'django.contrib.auth',
   'django.contrib.contenttypes',
   'django.contrib.sessions',
   'django.contrib.messages',
   'django.contrib.staticfiles',
   'myapp',
)
```

Creating forms in Django, is really similar to creating a model. Here again, we just need to inherit from Django class and the class attributes will be the form fields. Let's add a forms.py file in myapp folder to contain our app forms. We will create a login form.

myapp/forms.py

```
#-*- coding: utf-8 -*-
from django import forms
class LoginForm(forms.Form):
   user = forms.CharField(max_length = 100)
   password = forms.CharField(widget = forms.PasswordInput())
```

As seen above, the field type can take "widget" argument for html rendering; in our case, we want the password to be hidden, not displayed. Many others widget are present in Django: DateInput for dates, CheckboxInput for checkboxes, etc.

Using Form in a View

There are two kinds of HTTP requests, GET and POST. In Django, the request object passed as parameter to your view has an attribute called "method" where the type of the request is set, and all data passed via POST can be accessed via the request.POST dictionary.

Let's create a login view in our myapp/views.py −

```
#-*- coding: utf-8 -*-
from myapp.forms import LoginForm
def login(request):
   username = "not logged in"
   if request.method == "POST":
      #Get the posted form
      MyLoginForm = LoginForm(request.POST)
      if MyLoginForm.is_valid():
         username = MyLoginForm.cleaned_data['username']
   else:
      MyLoginForm = Loginform()
   return render(request, 'loggedin.html', {"username" : username})
```

The view will display the result of the login form posted through the loggedin.html. To test it, we will first need the login form template. Let's call it login.html.

```
<html>
   <body>
      <form name = "form" action = "{% url "myapp.views.login" %}"
         method = "POST" >{% csrf_token %}

         <div style = "max-width:470px;">
            <center>
               <input type = "text" style = "margin-left:20%;"
                  placeholder = "Identifiant" name = "username" />
            </center>
         </div>
         <br>
         <div style = "max-width:470px;">
            <center>
               <input type = "password" style = "margin-left:20%;"
                  placeholder = "password" name = "password" />
            </center>
         </div>
         <br>

         <div style = "max-width:470px;">
            <center>
               <button style = "border:0px; background-color:#4285F4; margin-top:8%;
                  height:35px; width:80%;margin-left:19%;" type = "submit"
                  value = "Login" >
                  <strong>Login</strong>
               </button>
            </center>
         </div>
      </form>
   </body>
```

```
</html>
```

The template will display a login form and post the result to our login view above. You have probably noticed the tag in the template, which is just to prevent Cross-site Request Forgery (CSRF) attack on your site.

```
{% csrf_token %}
```

Once we have the login template, we need the loggedin.html template that will be rendered after form treatment.

```
<html>
   <body>
      You are : <strong>{{username}}</strong>
   </body>
</html>
```

Now, we just need our pair of URLs to get started: myapp/urls.py

```
from django.conf.urls import patterns, url
from django.views.generic import TemplateView
urlpatterns = patterns('myapp.views',
   url(r'^connection/',TemplateView.as_view(template_name = 'login.html')),
   url(r'^login/', 'login', name = 'login'))
```

When accessing "/myapp/connection", we will get the following login.html template rendered

Setting Up Sessions

In Django, enabling session is done in your project settings.py, by adding some lines to the MIDDLEWARE_CLASSES and the INSTALLED_APPS options. This should be done while creating the project, but it's always good to know, so MIDDLEWARE_CLASSES should have

```
'django.contrib.sessions.middleware.SessionMiddleware'
```

And INSTALLED_APPS should have −

```
'django.contrib.sessions'
```

By default, Django saves session information in database (django_session table or collection), but you can configure the engine to store information using other ways like: in file or in cache.

When session is enabled, every request (first argument of any view in Django) has a session (dict) attribute.

Let's create a simple sample to see how to create and save sessions. We have built a simple login system before (see Django form processing chapter and Django Cookies Handling chapter). Let us save the username in a cookie so, if not signed out, when accessing our login page you won’t see the login form. Basically, let's make our login system we used in Django Cookies handling more secure, by saving cookies server side.

For this, first lets change our login view to save our username cookie server side −

```
def login(request):
   username = 'not logged in'
   if request.method == 'POST':
      MyLoginForm = LoginForm(request.POST)
      if MyLoginForm.is_valid():
         username = MyLoginForm.cleaned_data['username']
         request.session['username'] = username
      else:
         MyLoginForm = LoginForm()
   return render(request, 'loggedin.html', {"username" : username}
```

Then let us create formView view for the login form, where we won’t display the form if cookie is set −

```
def formView(request):
   if request.session.has_key('username'):
      username = request.session['username']
      return render(request, 'loggedin.html', {"username" : username})
   else:
      return render(request, 'login.html', {})
```

Now let us change the url.py file to change the url so it pairs with our new view −

```
from django.conf.urls import patterns, url
from django.views.generic import TemplateView
urlpatterns = patterns('myapp.views',
   url(r'^connection/','formView', name = 'loginform'),
   url(r'^login/', 'login', name = 'login'))
```

When accessing /myapp/connection, you will get to see the following page

CHAPTER 6: SYSTEM STUDY

6.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are:

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

6.1.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

6.1.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

6.1.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 7: SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

7.1 TYPES OF TESTS

7.1.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

7.1.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

7.1.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	:	identified classes of valid input must be accepted.
Invalid Input	:	identified classes of invalid input must be rejected.

Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

7.1.4 SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

7.1.5 WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

7.1.6 BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

7.1.7 UNIT TESTING

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

7.2 TEST STRATEGY AND APPROACH

Field testing will be performed manually and functional tests will be written in detail.

7.2.1 TEST OBJECTIVES

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

7.2.2 INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

7.2.3 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

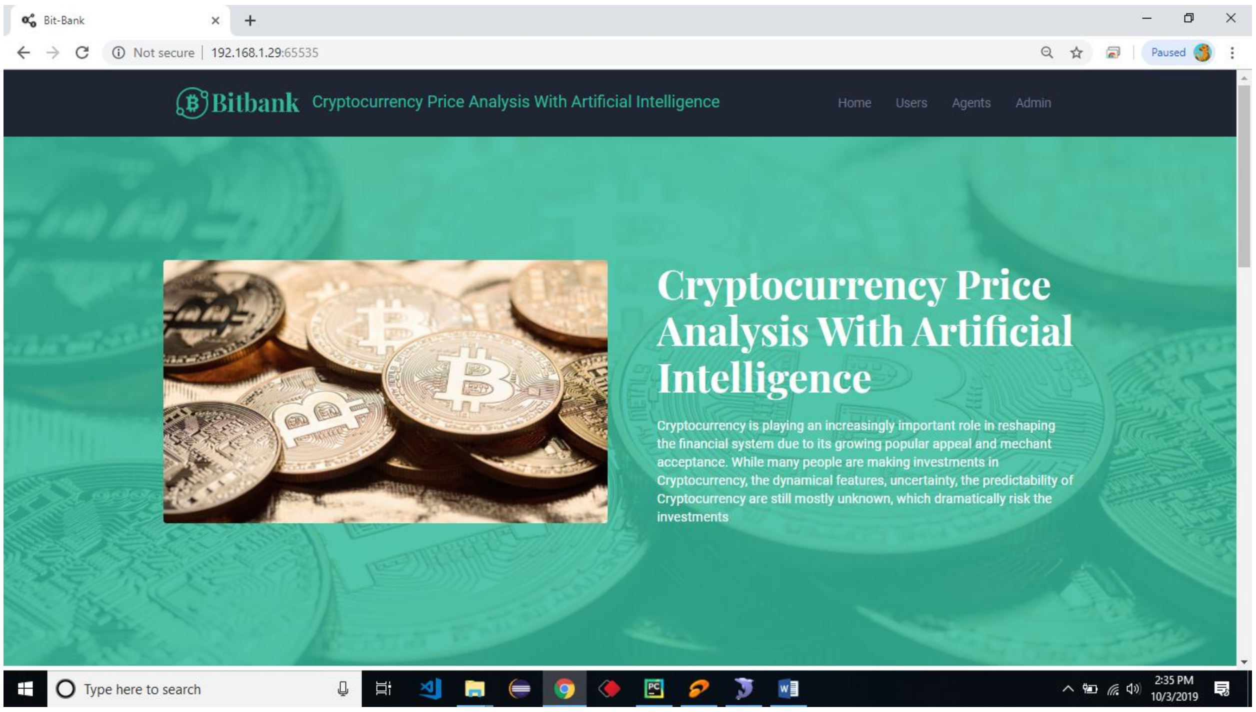
Test Results: All the test cases mentioned above passed successfully. No defects encountered.

7.3 SAMPLE TEST CASES

S.no	Test Case	Excepted Result	Result	Remarks (IF fails)
1	User REGISTERED	If user registration successfully.	Pass	If user is not registered.
2	Agent REGISTERED	If agent registration successfully.	Pass	If agent is not registered.
3	ADMIN	user rights will be accepted here.	Pass	If user is not registered.
4	ADMIN	agent rights will be accepted here.	Pass	If agent is not registered.
5	user LOGIN	If user name and password is correct then it will getting valid page.	Pass	If user name or password is not correct.
6	agent LOGIN	If agent name and password is correct then it will getting valid page.	Pass	If agent name or password is not correct.
7	Agent buying crypto currency from admin	If agent is correct then it will getting valid page.	Pass	If sale crypto currencies are not available.
8	User buying crypto currency from agent	If user is correct then it will getting valid page	Pass	If sale crypto currencies are not available

CHAPTER 8: RESULTS

SAMPLE SCREENS

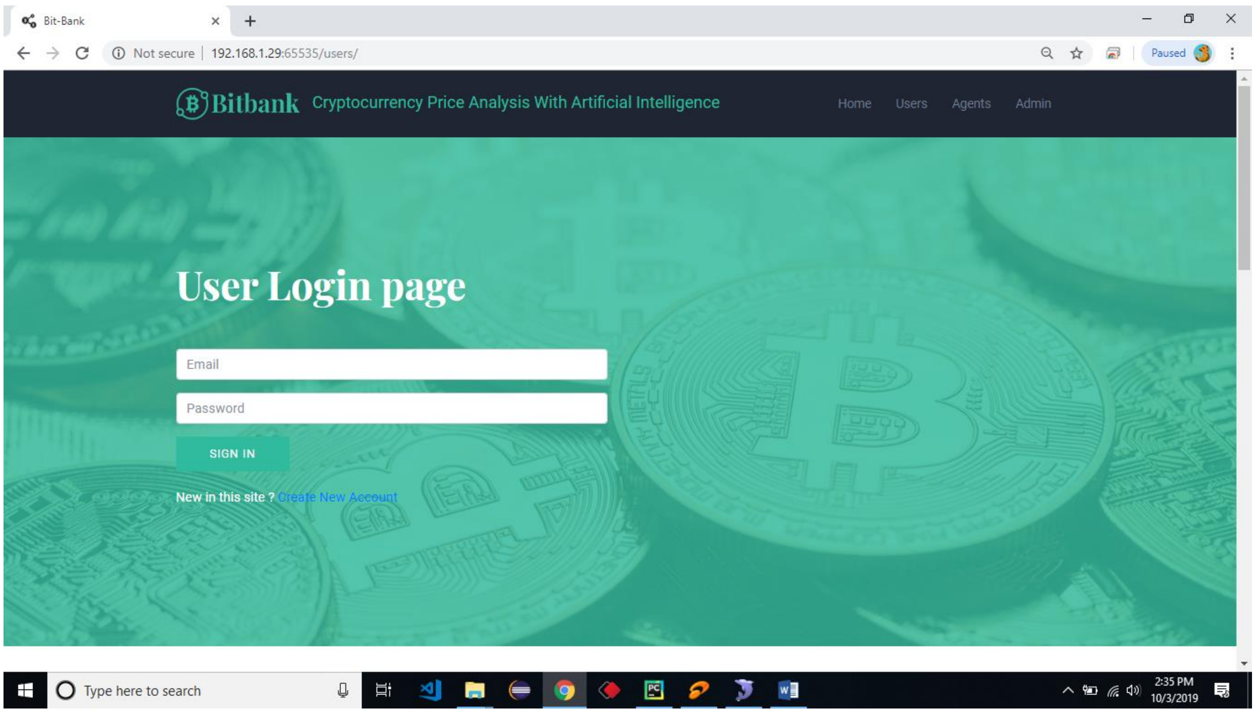


Bit-Bank
Not secure | 192.168.1.29:65535
Bitbank
Cryptocurrency Price Analysis With Artificial Intelligence
Home
Users
Agents
Admin

Cryptocurrency Price Analysis With Artificial Intelligence

Cryptocurrency is playing an increasingly important role in reshaping the financial system due to its growing popular appeal and mechant acceptance. While many people are making investments in Cryptocurrency, the dynamical features, uncertainty, the predictability of Cryptocurrency are still mostly unknown, which dramatically risk the investments

Fig 11: Main Home Page



Bit-Bank
Not secure | 192.168.1.29:65535/users/
Bitbank
Cryptocurrency Price Analysis With Artificial Intelligence
Home
Users
Agents
Admin

User Login page

Email

Password

SIGN IN

New in this site ? Create New Account

Type here to search
2:35 PM
10/3/2019

Fig 12: User Register Page

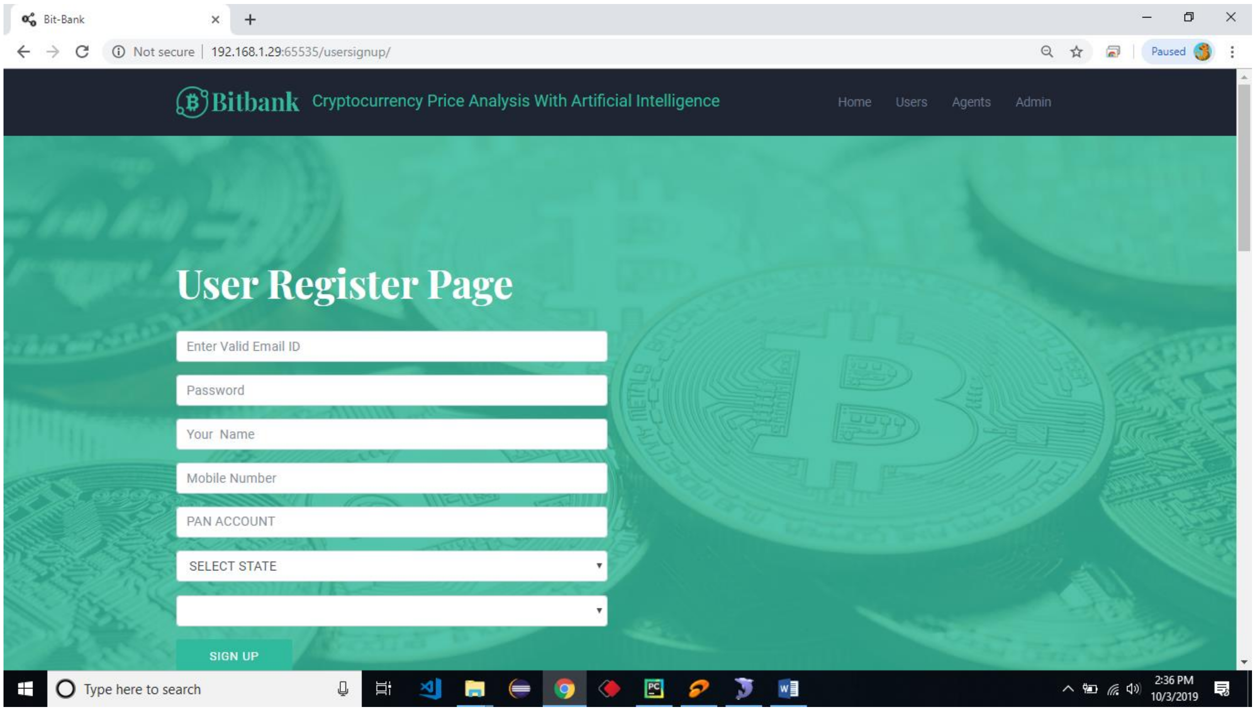


Fig 13: User Registration Form

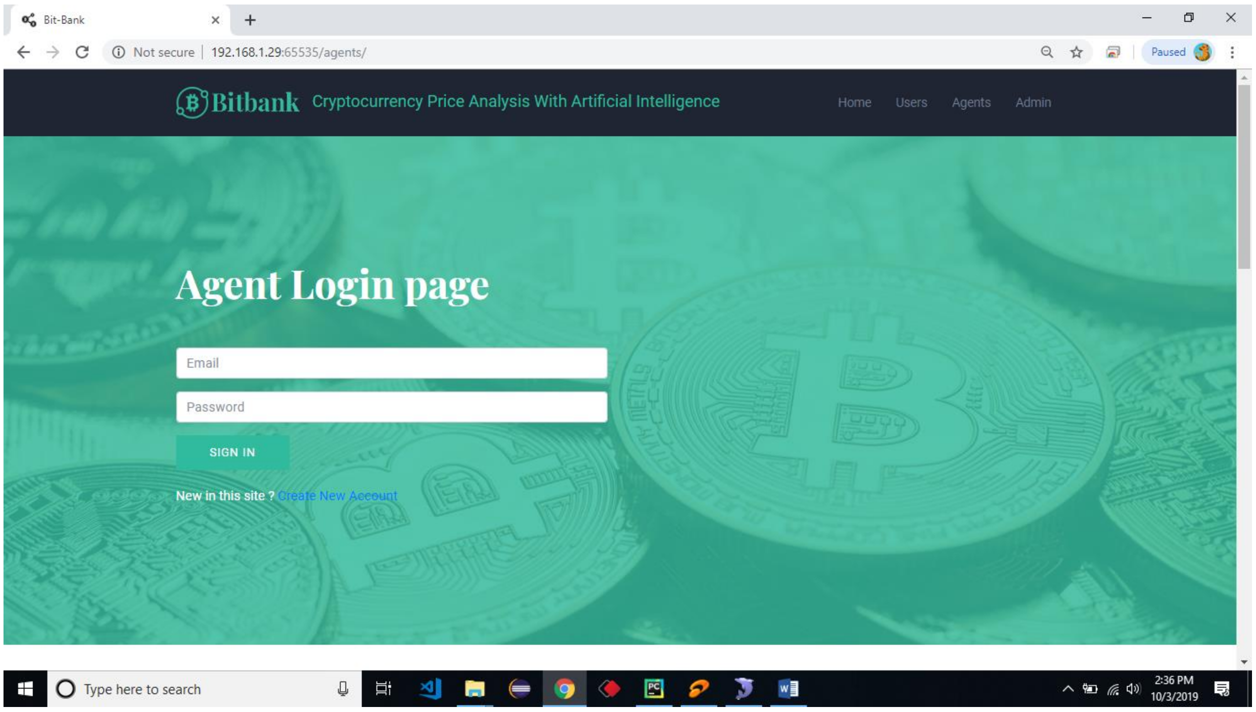

Bit-Bank
Not secure | 192.168.1.29:65535/agents/
Bitbank Cryptocurrency Price Analysis With Artificial Intelligence
Home Users Agents Admin
Agent Login page
Email
Password
SIGN IN
New in this site ? Create New Account
Type here to search
2:36 PM 10/3/2019

Fig 14: Agent Login Page

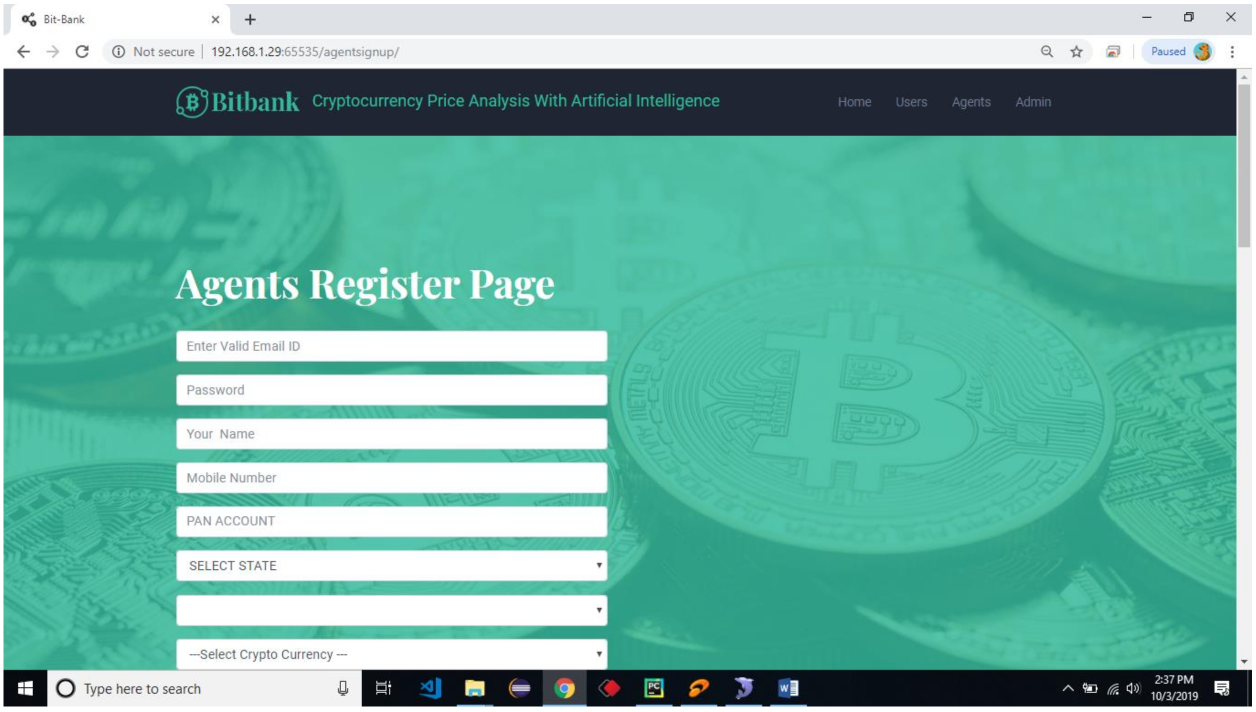


Fig 15: Agent Register Page

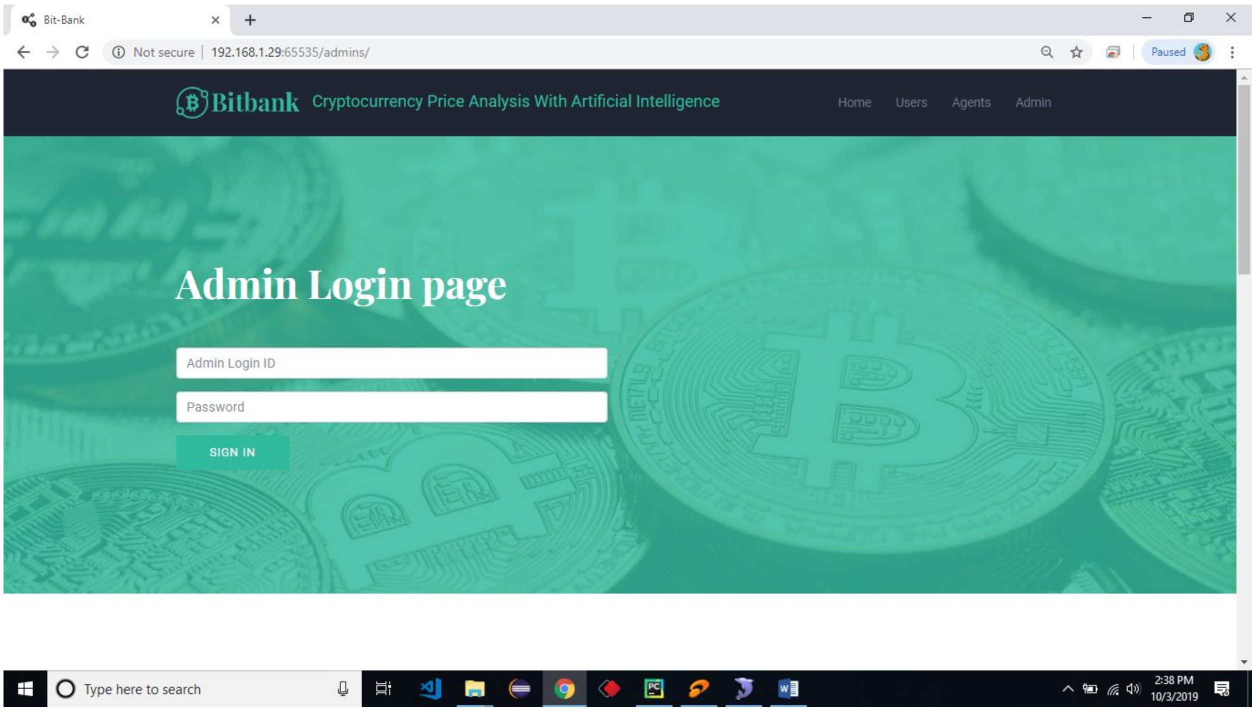
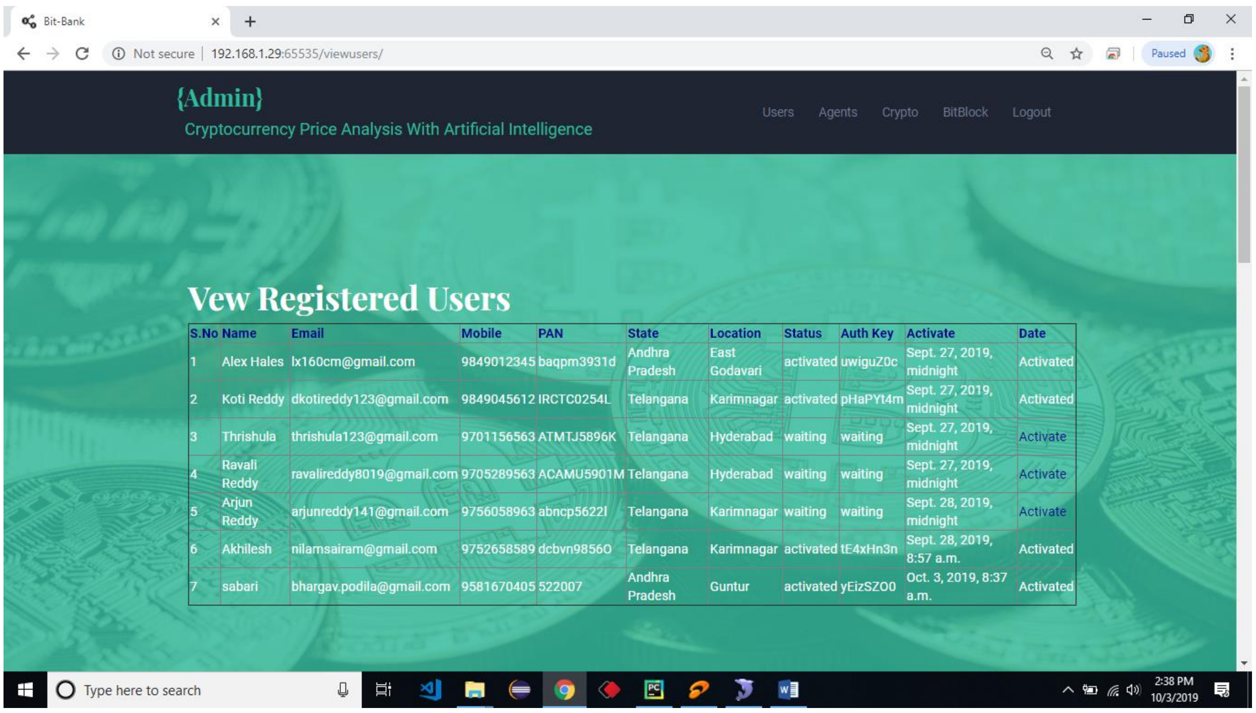


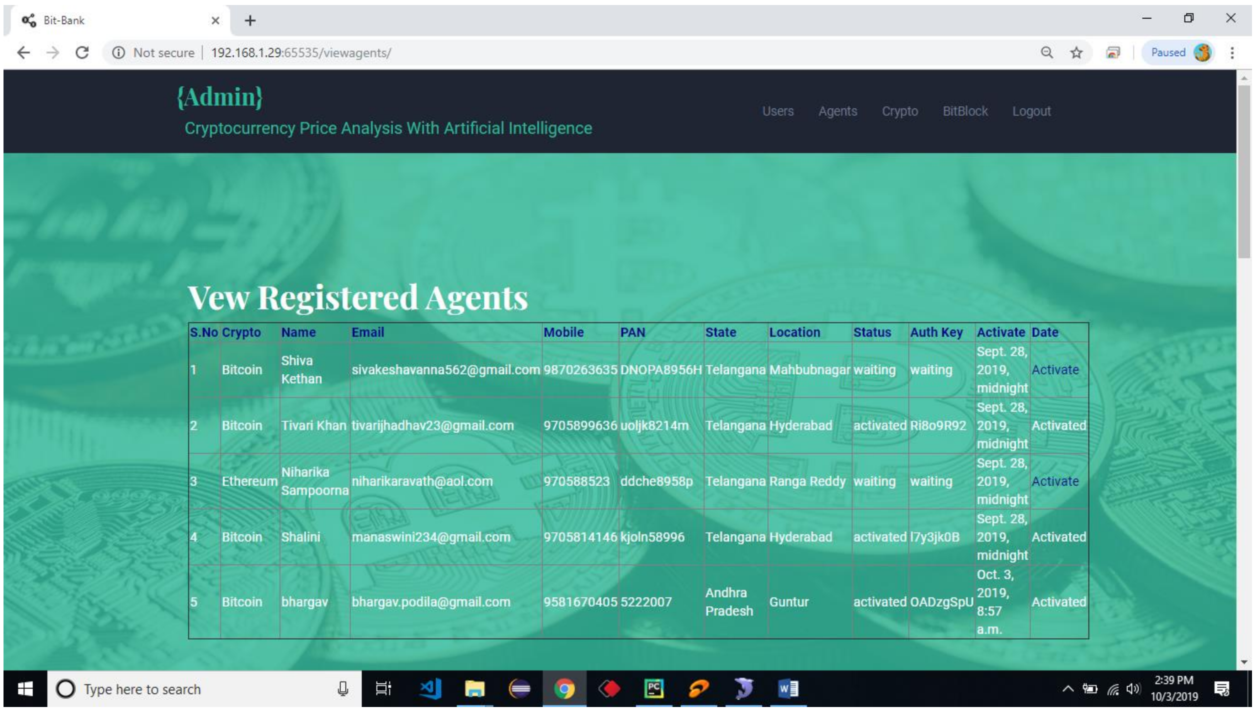
Fig 16: Admin Login Page


Bit-Bank
Not secure | 192.168.1.29:65535/viewusers/
Paused
{Admin}
Cryptocurrency Price Analysis With Artificial Intelligence
Users
Agents
Crypto
BitBlock
Logout
Type here to search
2:38 PM
10/3/2019

Vew Registered Users

S.No	Name	Email	Mobile	PAN	State	Location	Status	Auth Key	Activate	Date
1	Alex Hales	lx160cm@gmail.com	9849012345	baqpm3931d	Andhra Pradesh	East Godavari	activated	uwiguZ0c	Sept. 27, 2019, midnight	Activated
2	Koti Reddy	dkotireddy123@gmail.com	9849045612	IRCTC0254L	Telangana	Karimnagar	activated	pHaPYt4m	Sept. 27, 2019, midnight	Activated
3	Thrishula	thrishula123@gmail.com	9701156563	ATMTJ5896K	Telangana	Hyderabad	waiting	waiting	Sept. 27, 2019, midnight	Activate
4	Ravali Reddy	ravalireddy8019@gmail.com	9705289563	ACAMU5901M	Telangana	Hyderabad	waiting	waiting	Sept. 27, 2019, midnight	Activate
5	Arjun Reddy	arjunreddy141@gmail.com	9756058963	abncp5622l	Telangana	Karimnagar	waiting	waiting	Sept. 28, 2019, midnight	Activate
6	Akhilesh	nilamsairam@gmail.com	9752658589	dcbvn98560	Telangana	Karimnagar	activated	tE4xHn3n	Sept. 28, 2019, 8:57 a.m.	Activated
7	sabari	bhargav.podila@gmail.com	9581670405	522007	Andhra Pradesh	Guntur	activated	yEizSZO0	Oct. 3, 2019, 8:37 a.m.	Activated

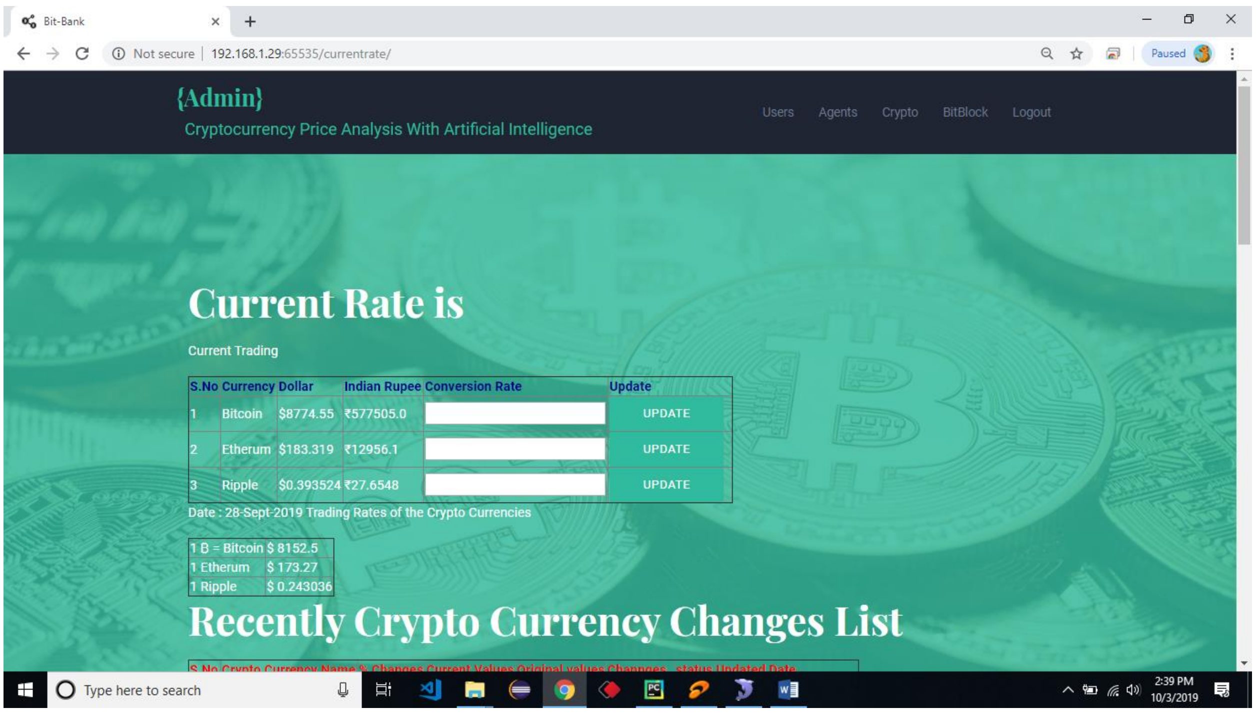
Fig 17: Admin Activate Users


Bit-Bank
Not secure | 192.168.1.29:65535/viewagents/
Paused
{Admin}
Cryptocurrency Price Analysis With Artificial Intelligence
Users
Agents
Crypto
BitBlock
Logout
Type here to search
2:39 PM
10/3/2019

Vew Registered Agents

S.No	Crypto	Name	Email	Mobile	PAN	State	Location	Status	Auth Key	Activate	Date
1	Bitcoin	Shiva Kethan	sivakeshavanna562@gmail.com	9870263635	DNOPA8956H	Telangana	Mahbubnagar	waiting	waiting	Sept. 28, 2019, midnight	Activate
2	Bitcoin	Tivari Khan	tivarijhadhav23@gmail.com	9705899636	uoljk8214m	Telangana	Hyderabad	activated	Ri8o9R92	Sept. 28, 2019, midnight	Activated
3	Ethereum	Niharika Sampoorna	niharikaravath@aol.com	970588523	ddche8958p	Telangana	Ranga Reddy	waiting	waiting	Sept. 28, 2019, midnight	Activate
4	Bitcoin	Shalini	manaswini234@gmail.com	9705814146	kjoln58996	Telangana	Hyderabad	activated	l7y3jk0B	Sept. 28, 2019, midnight	Activated
5	Bitcoin	bhargav	bhargav.podila@gmail.com	9581670405	5222007	Andhra Pradesh	Guntur	activated	OADzgSpU	Oct. 3, 2019, 8:57 a.m.	Activated

Fig 18: Admin Activate Agents



Bit-Bank

Not secure | 192.168.1.29:65535/currentrate/

{Admin}

Cryptocurrency Price Analysis With Artificial Intelligence

Users Agents Crypto BitBlock Logout

Current Rate is

Current Trading

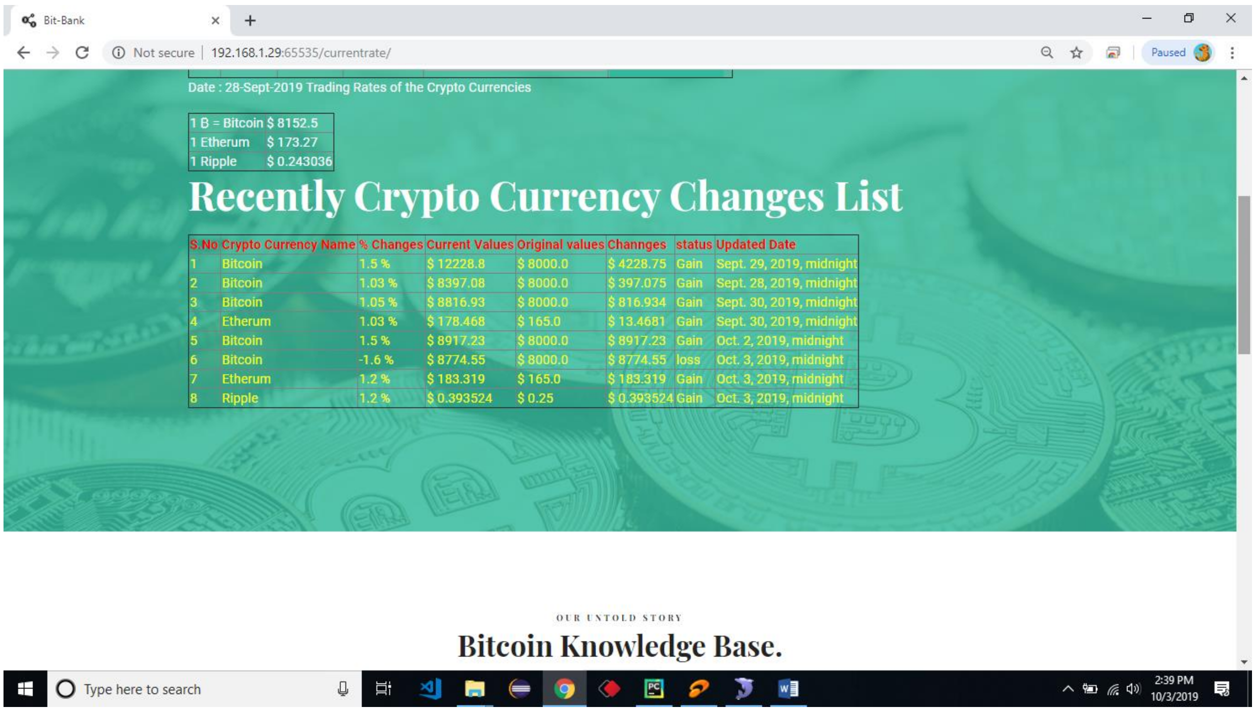
S.No	Currency	Dollar	Indian Rupee	Conversion Rate	Update
1	Bitcoin	$8774.55	₹577505.0		UPDATE
2	Etherum	$183.319	₹12956.1		UPDATE
3	Ripple	$0.393524	₹27.6548		UPDATE

Date : 28-Sept-2019 Trading Rates of the Crypto Currencies

1 Ƀ = Bitcoin	$ 8152.5
1 Etherum	$ 173.27
1 Ripple	$ 0.243036

Recently Crypto Currency Changes List

Fig 19: Current Price and Update



Bit-Bank

Not secure | 192.168.1.29:65535/currentrate/

Date : 28-Sept-2019 Trading Rates of the Crypto Currencies

1 Ƀ = Bitcoin	$ 8152.5
1 Etherum	$ 173.27
1 Ripple	$ 0.243036

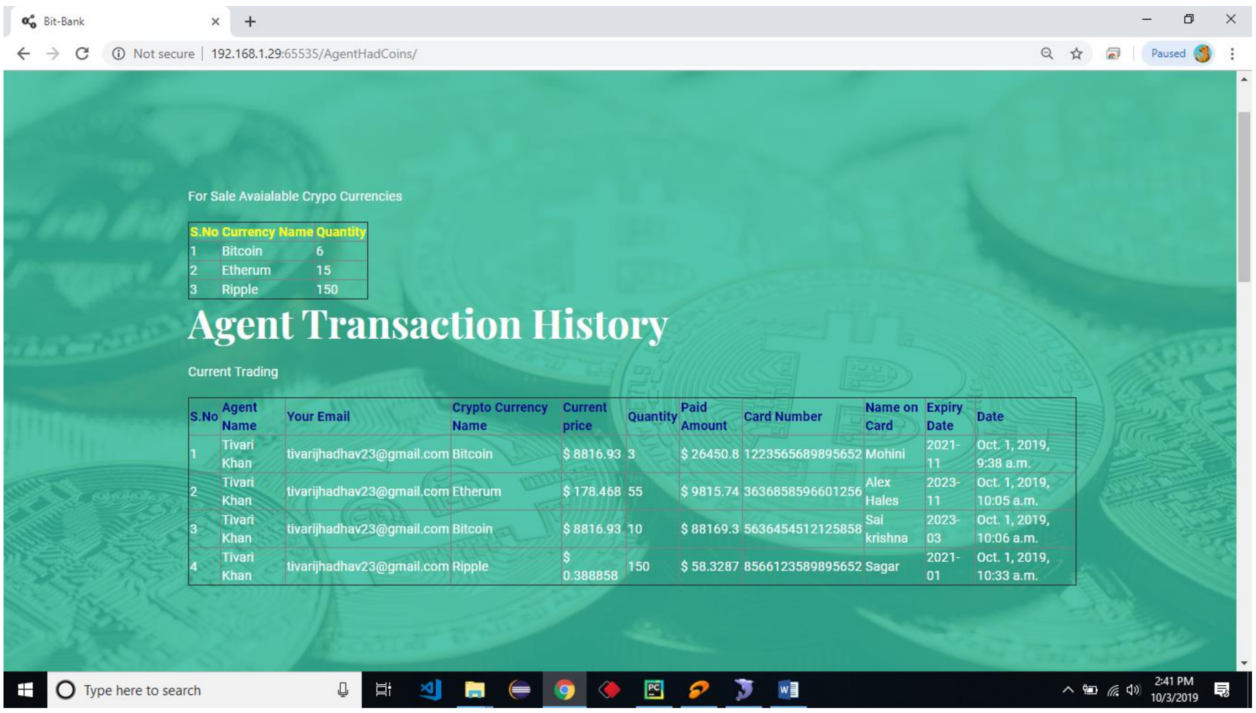
Recently Crypto Currency Changes List

S.No	Crypto Currency Name	% Changes	Current Values	Original values	Channges	status	Updated Date
1	Bitcoin	1.5 %	$ 12228.8	$ 8000.0	$ 4228.75	Gain	Sept. 29, 2019, midnight
2	Bitcoin	1.03 %	$ 8397.08	$ 8000.0	$ 397.075	Gain	Sept. 28, 2019, midnight
3	Bitcoin	1.05 %	$ 8816.93	$ 8000.0	$ 816.934	Gain	Sept. 30, 2019, midnight
4	Etherum	1.03 %	$ 178.468	$ 165.0	$ 13.4681	Gain	Sept. 30, 2019, midnight
5	Bitcoin	1.5 %	$ 8917.23	$ 8000.0	$ 8917.23	Gain	Oct. 2, 2019, midnight
6	Bitcoin	-1.6 %	$ 8774.55	$ 8000.0	$ 8774.55	loss	Oct. 3, 2019, midnight
7	Etherum	1.2 %	$ 183.319	$ 165.0	$ 183.319	Gain	Oct. 3, 2019, midnight
8	Ripple	1.2 %	$ 0.393524	$ 0.25	$ 0.393524	Gain	Oct. 3, 2019, midnight

OUR UNTOLD STORY

Bitcoin Knowledge Base.

Fig 20: Crypto Update History



For Sale Avaialable Crypo Currencies

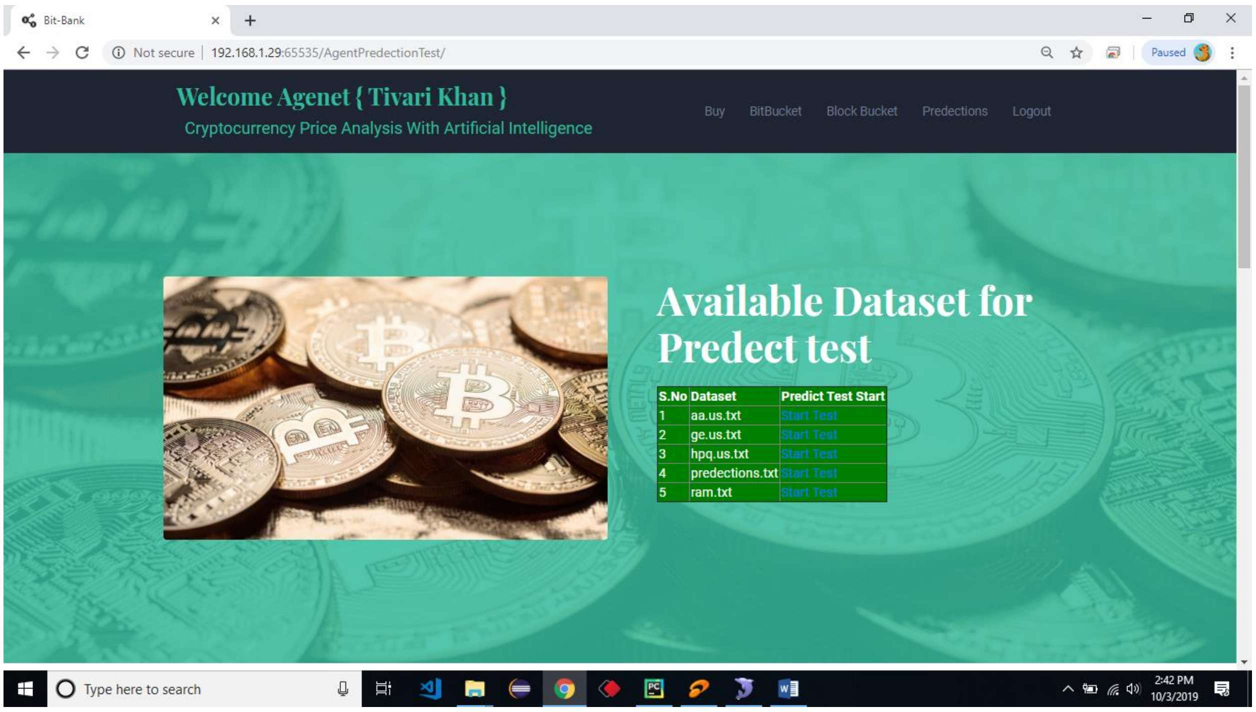
S.No	Currency Name	Quantity
1	Bitcoin	6
2	Etherum	15
3	Ripple	150

Agent Transaction History

Current Trading

S.No	Agent Name	Your Email	Crypto Currency Name	Current price	Quantity	Paid Amount	Card Number	Name on Card	Expiry Date	Date
1	Tivari Khan	tivarijhadhav23@gmail.com	Bitcoin	$ 8816.93	3	$ 26450.8	1223565689895652	Mohini	2021-11	Oct. 1, 2019, 9:38 a.m.
2	Tivari Khan	tivarijhadhav23@gmail.com	Etherum	$ 178.468	55	$ 9815.74	3636858596601256	Alex Hales	2023-11	Oct. 1, 2019, 10:05 a.m.
3	Tivari Khan	tivarijhadhav23@gmail.com	Bitcoin	$ 8816.93	10	$ 88169.3	5636454512125858	Sai krishna	2023-03	Oct. 1, 2019, 10:06 a.m.
4	Tivari Khan	tivarijhadhav23@gmail.com	Ripple	$ 0.388858	150	$ 58.3287	8566123589895652	Sagar	2021-01	Oct. 1, 2019, 10:33 a.m.

Fig 21: Agent Transaction History



Welcome Agenet { Tivari Khan }

Cryptocurrency Price Analysis With Artificial Intelligence

Available Dataset for Predect test

S.No	Dataset	Predict Test Start
1	aa.us.txt	Start Test
2	ge.us.txt	Start Test
3	hpq.us.txt	Start Test
4	predections.txt	Start Test
5	ram.txt	Start Test

Fig 22: Agent View Predictions Dataset for Test

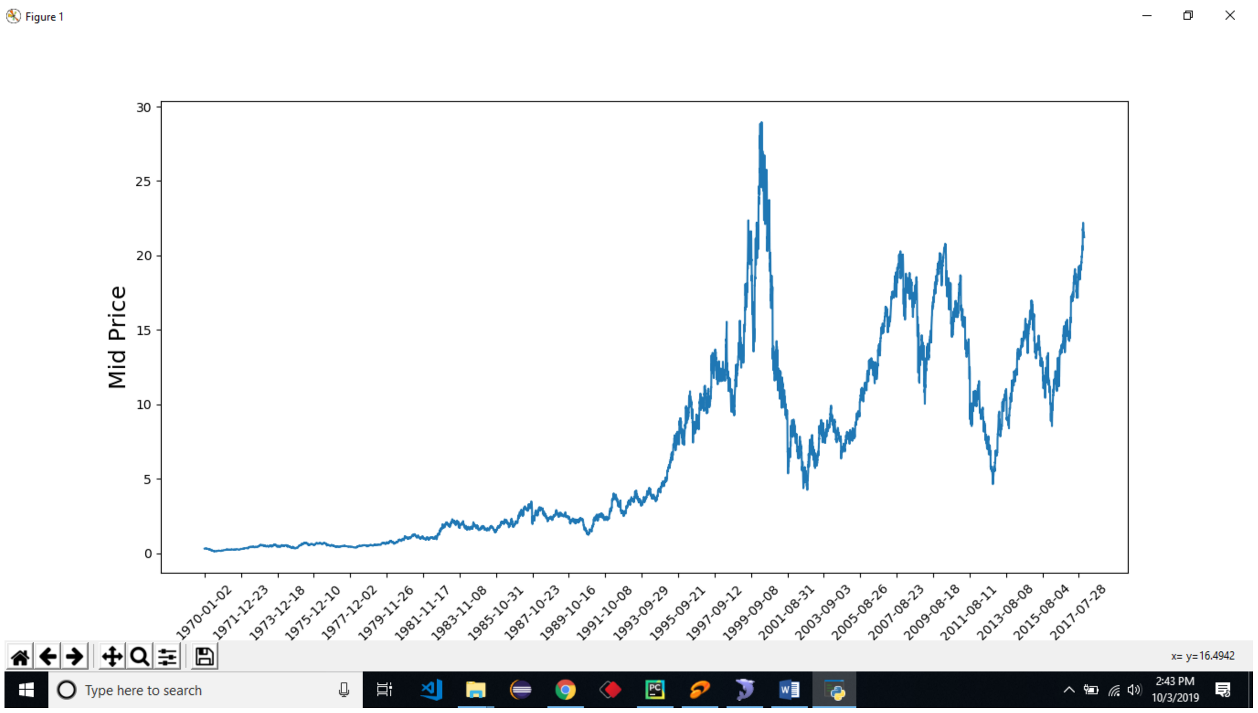


Fig 23: Dataset Analysis

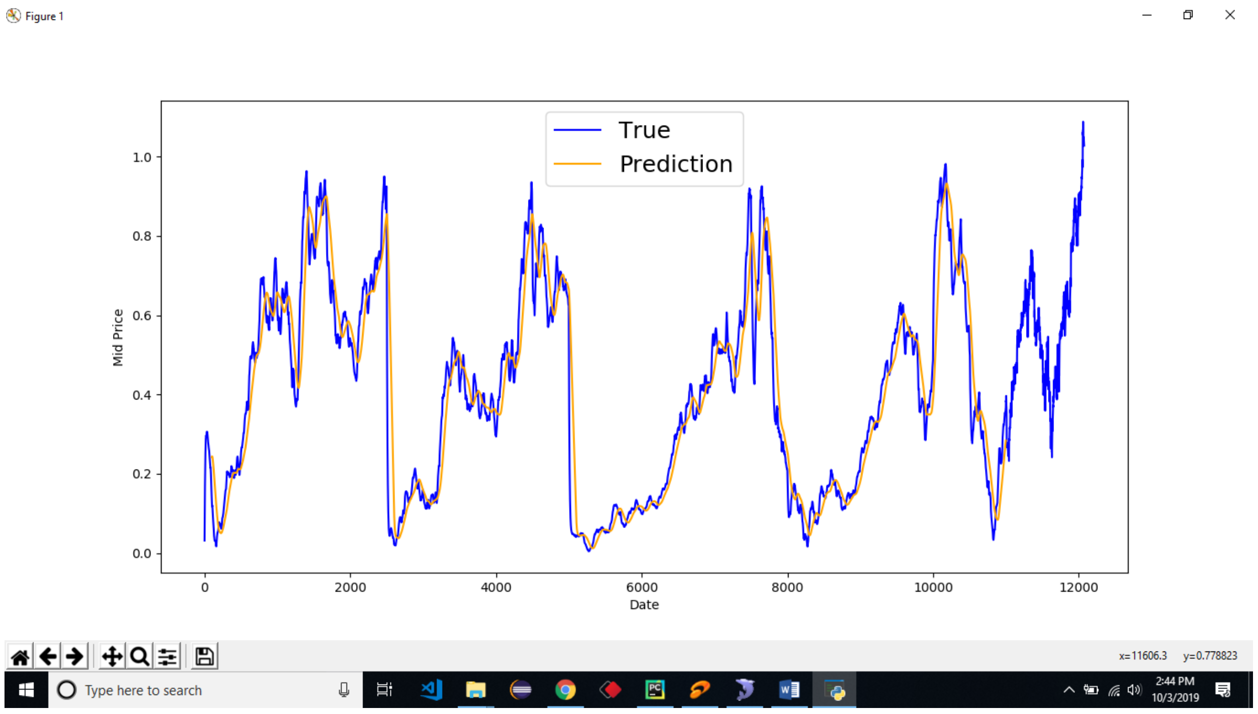


Fig 24: True Predictions

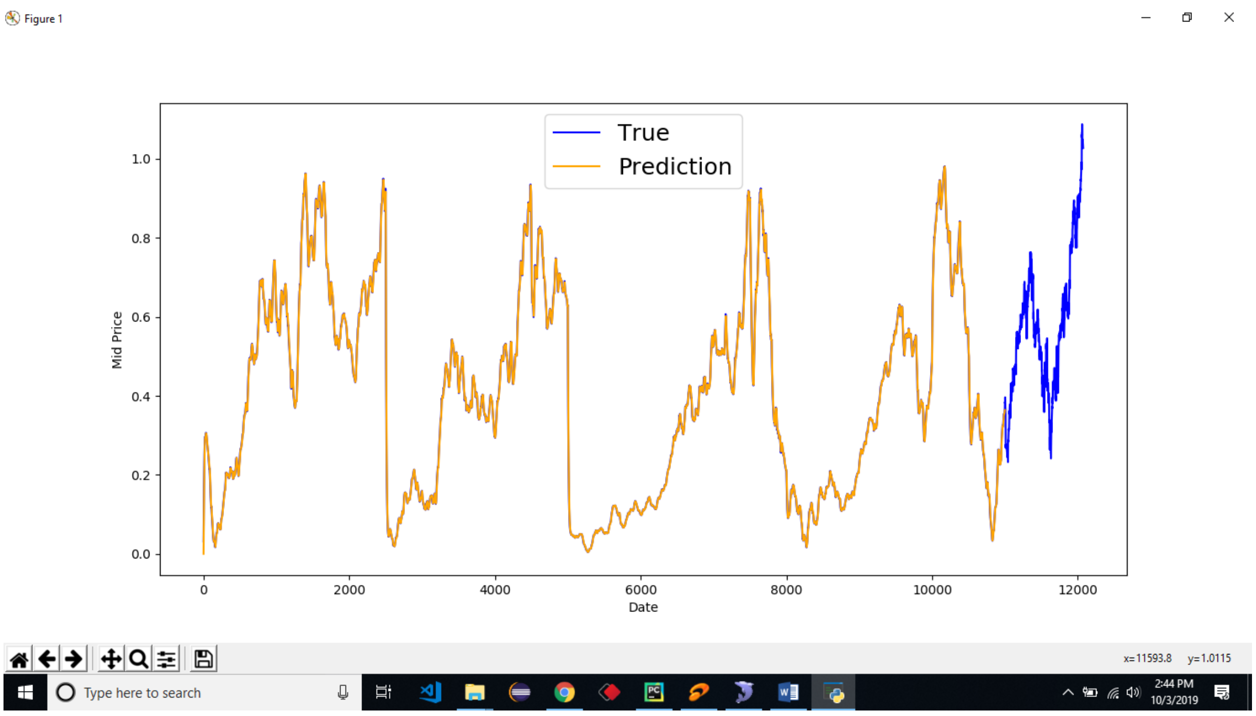
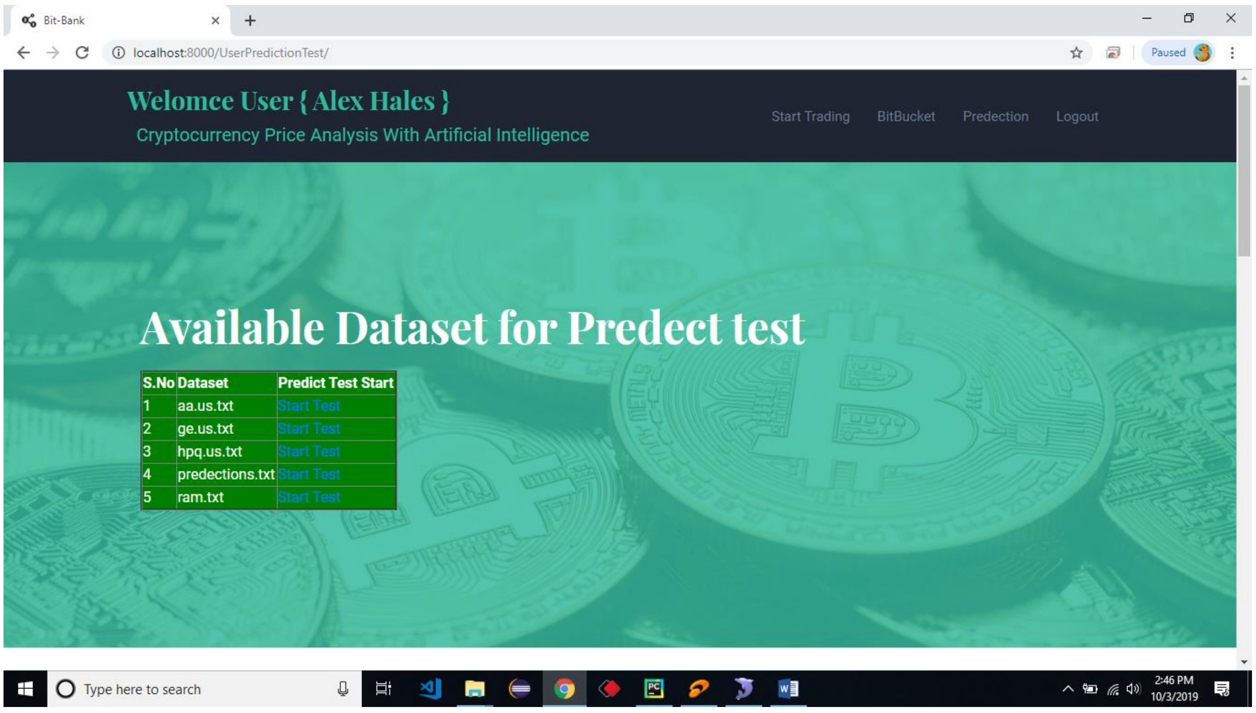


Fig 25: Predictions



Welomce User { Alex Hales }

Cryptocurrency Price Analysis With Artificial Intelligence

Available Dataset for Predect test

S.No	Dataset	Predict Test Start
1	aa.us.txt	Start Test
2	ge.us.txt	Start Test
3	hpq.us.txt	Start Test
4	predections.txt	Start Test
5	ram.txt	Start Test

Fig 26: User Can Test the Predictions

CHAPTER 9
CONCLUSION AND FUTURE ENHANCEMENT

9.1 CONCLUSION

Cryptocurrency, such as Bitcoin, has established itself as the leading role of decentralization. There are a large number of cryptocurrencies sprang up after Bitcoin such as Ethereum and Ripple. Because of the significant uncertainty in its prices, many people hold them as a means of speculation. Therefore, it is critically important to understand the internal features and predictability of those cryptocurrencies. In this study, we use two distinct artificial intelligence frameworks, namely, fully-connected Artificial Neural Network (ANN) and Long-Short-Term-Memory (LSTM) to analyze and predict the price dynamics of Bitcoin, Ethereum, and Ripple. We showed that the ANN and LSTM models are comparable and both reasonably well enough in price prediction, although the internal structures are different. Then we further analyze the influence of historical memory on model prediction. We find that ANN tends to rely more on long-term history while LSTM tends to rely more on short-term dynamics, which indicate the efficiency of LSTM to utilize useful information hidden in historical memory is stronger than ANN. However, given enough historical information ANN can achieve a similar accuracy, compared with LSTM. This study provides a unique demonstration that Cryptocurrency market price is predictable. However, the explanation of the predictability could vary depending on the nature of the involved machine-learning model.

9.2 FUTURE ENHANCEMENT

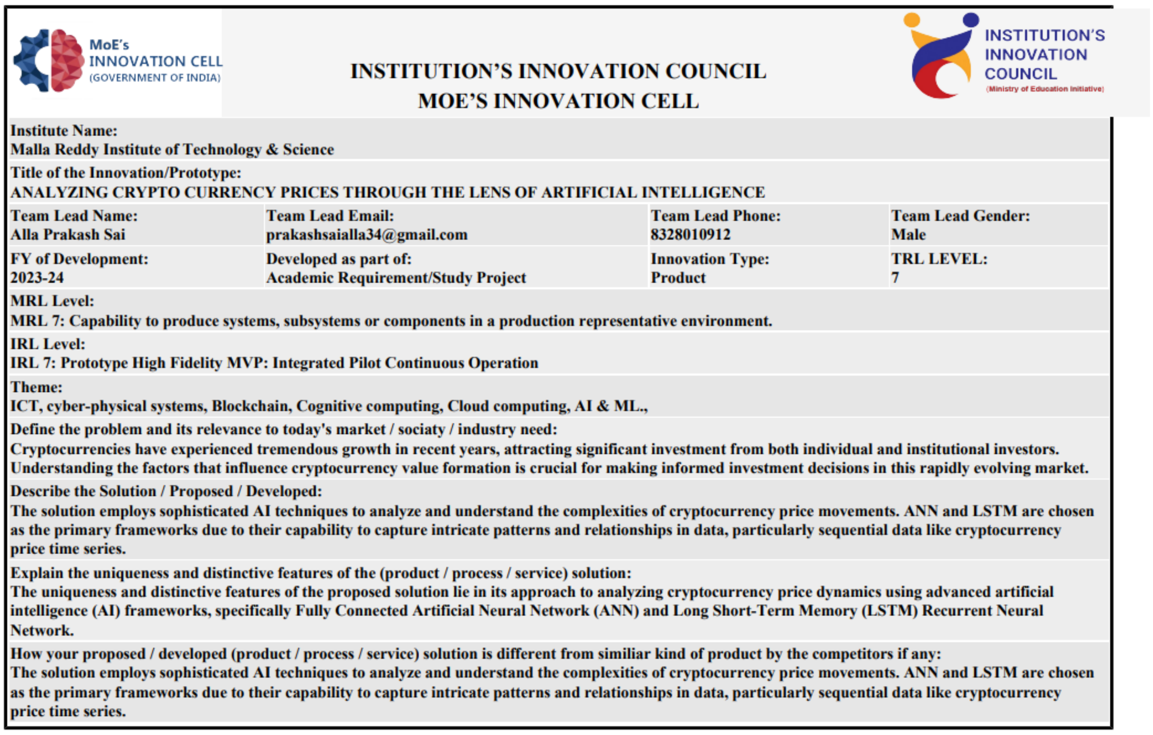
In the future, enhancements to the proposed system for cryptocurrency price analysis using advanced AI frameworks could significantly augment its capabilities. Integration of additional data sources like trading volumes, sentiment indicators, and macroeconomic factors would provide a more comprehensive understanding of market dynamics, thus refining the accuracy of price predictions. The incorporation of reinforcement learning techniques could further optimize decision-making processes and trading strategies, enabling the system to adapt and learn from real-time feedback. Development of explainable AI models would improve transparency, allowing users to better comprehend the rationale behind the system's recommendations. Real-time analysis capabilities would empower users to capitalize on fleeting market opportunities and swiftly respond to changing conditions. Integration with trading platforms would streamline execution of trading strategies based on the system's insights, enhancing trading performance.

CHAPTER 10: BIBLIOGRAPHY

[1] Greaves, A., & Au, B. (2015). Using the bitcoin transaction graph to predict the price of bitcoin. No Data.
[2] Hayes, A. S. (2017). Cryptocurrency value formation: An empirical study leading to a cost of production model for valuing bitcoin. Telematics and Informatics, 34(7), 1308-1321.
[3] Shah, D., & Zhang, K. (2014, September). Bayesian regression and Bitcoin. In Communication, Control, and Computing (Allerton), 2014 52nd Annual Allerton Conference on (pp. 409-414). IEEE.
[4] Indra N I, Yassin I M, Zabidi A, Rizman Z I. Non-linear autoregressive with exogenous input (mrx) bitcoin price prediction model using so-optimized parameters and moving average technical indicators. J. Fundam. Appl. Sci., 2017, 9(3S), 791-808`
[5] Adebiyi AA, Ayo C K, Adebiyi MO, Otokiti SO. Stock price prediction using a neural network with hybridized market indicators. Journal of Emerging Trends in Computing and Information Sciences, 2012, 3(1):1-9
[6] Adebiyi AA, Ayo C K, Adebiyi MO, Otokiti SO. Stock price prediction using a neural network with hybridized market indicators. Journal of Emerging Trends in Computing and Information Sciences, 2012, 3(1):1-9
[7] Ariyo AA, Adewumi AO, Ayo CK. Stock price prediction using the ARIMA model. In UKSim-AMSS 16th IEEE International Conference on Computer Modelling and Simulation (UKSim), 2014, pp. 106-112
[8] Ron, D., & Shamir, A. (2013, April). Quantitative analysis of the full bitcoin transaction graph. In International Conference on Financial Cryptography and Data Security (pp. 6-24). Springer, Berlin, Heidelberg.
[9] H. White, "Economic prediction using neural networks: The case of ibm daily stock returns," in Neural Networks, 1988., IEEE International Conference on. IEEE, 1988, pp. 451–458
[10] Kaastra and M. Boyd, "Designing a neural network for forecasting financial and economic time series," Neurocomputing, vol. 10, no. 3, pp. 215–236, 1996.
[11] Cheung, Y. W., Chinn, M. D., & Pascual, A. G. (2005). Empirical exchange rate models of the nineties: Are any fit to survive? Journal of international money and finance, 24(7), 1150-1175.

CHAPTER 11

YUKTI INNOVATION CERTIFICATE


MoE's INNOVATION CELL (GOVERNMENT OF INDIA)

INSTITUTION'S INNOVATION COUNCIL
MOE'S INNOVATION CELL

INSTITUTION'S INNOVATION COUNCIL (Ministry of Education Initiative)

Institute Name:
Malla Reddy Institute of Technology & Science

Title of the Innovation/Prototype:
ANALYZING CRYPTO CURRENCY PRICES THROUGH THE LENS OF ARTIFICIAL INTELLIGENCE

Team Lead Name: Alla Prakash Sai	**Team Lead Email:** prakashsaialla34@gmail.com	**Team Lead Phone:** 8328010912	**Team Lead Gender:** Male
FY of Development: 2023-24	**Developed as part of:** Academic Requirement/Study Project	**Innovation Type:** Product	**TRL LEVEL:** 7

MRL Level:
MRL 7: Capability to produce systems, subsystems or components in a production representative environment.

IRL Level:
IRL 7: Prototype High Fidelity MVP: Integrated Pilot Continuous Operation

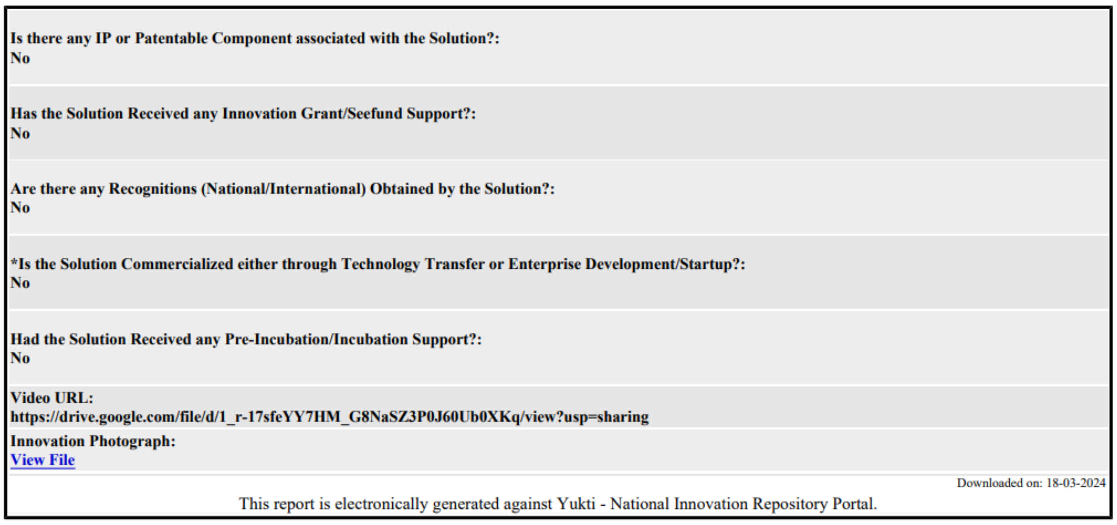
Theme:
ICT, cyber-physical systems, Blockchain, Cognitive computing, Cloud computing, AI & ML.,

Define the problem and its relevance to today's market / sociaty / industry need:
Cryptocurrencies have experienced tremendous growth in recent years, attracting significant investment from both individual and institutional investors. Understanding the factors that influence cryptocurrency value formation is crucial for making informed investment decisions in this rapidly evolving market.

Describe the Solution / Proposed / Developed:
The solution employs sophisticated AI techniques to analyze and understand the complexities of cryptocurrency price movements. ANN and LSTM are chosen as the primary frameworks due to their capability to capture intricate patterns and relationships in data, particularly sequential data like cryptocurrency price time series.

Explain the uniqueness and distinctive features of the (product / process / service) solution:
The uniqueness and distinctive features of the proposed solution lie in its approach to analyzing cryptocurrency price dynamics using advanced artificial intelligence (AI) frameworks, specifically Fully Connected Artificial Neural Network (ANN) and Long Short-Term Memory (LSTM) Recurrent Neural Network.

How your proposed / developed (product / process / service) solution is different from similiar kind of product by the competitors if any:
The solution employs sophisticated AI techniques to analyze and understand the complexities of cryptocurrency price movements. ANN and LSTM are chosen as the primary frameworks due to their capability to capture intricate patterns and relationships in data, particularly sequential data like cryptocurrency price time series.


Is there any IP or Patentable Component associated with the Solution?:
No

Has the Solution Received any Innovation Grant/Seefund Support?:
No

Are there any Recognitions (National/International) Obtained by the Solution?:
No

***Is the Solution Commercialized either through Technology Transfer or Enterprise Development/Startup?:**
No

Had the Solution Received any Pre-Incubation/Incubation Support?:
No

Video URL:
https://drive.google.com/file/d/1_r-17sfeYY7HM_G8NaSZ3P0J60Ub0XKq/view?usp=sharing

Innovation Photograph:
View File

Downloaded on: 18-03-2024

This report is electronically generated against Yukti - National Innovation Repository Portal.