NETAJI SUBHAS INSTITUTE OF TECHNOLOGY, BIHTA, PATNA

INTERNSHIP REPORT

ON

DECENTRALISED APPLICATION FOR CROP DISEASE PREDICTION AND INSURANCE

SUBMITTED IN THE PARTIAL FULFILLMENT OF THE REQUIREMENT

OF

BACHELOR OF TECHNOLOGY (COMPUTER SCIENCE AND ENGINEERING)

BY

ARYABHATTA KNOWLEDGE UNIVERSITY, PATNA, BIHAR

SUBMITTED BY:

Swadha Kumari (181029)

Karan Kumar (181040)

Priti Kumari (181038)

Vth sem

SESSION 2018-22

An ISO 9001:2008 Certified Institution

Approved by AICTE, New Delhi

Affiliated to Aryabhatta Knowledge University, Patna, Bihar

NETAJI SUBHAS INSTITUTE OF TECHNOLOGY, BIHTA, PATNA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SESSION- 2018-22

ACKNOWLEDGEMENT

The internship opportunity, we had with IIT Patna, was a great chance for learning and professional development. Therefore, we consider ourselves lucky and would like to express our deepest gratitude to Dr. Raju Halder, Assistant Professor, IIT Patna, who inspite of having a busy schedule, heartily guided and encouraged us throughout the internship.

We would also like to convey our deepest gratitude to Mr. Gopal Krishna, Assistant Professor, NSIT Patna, for taking part in useful decisions and giving necessary advice and guidance and also for arranging all facilities to make the training experience easier.

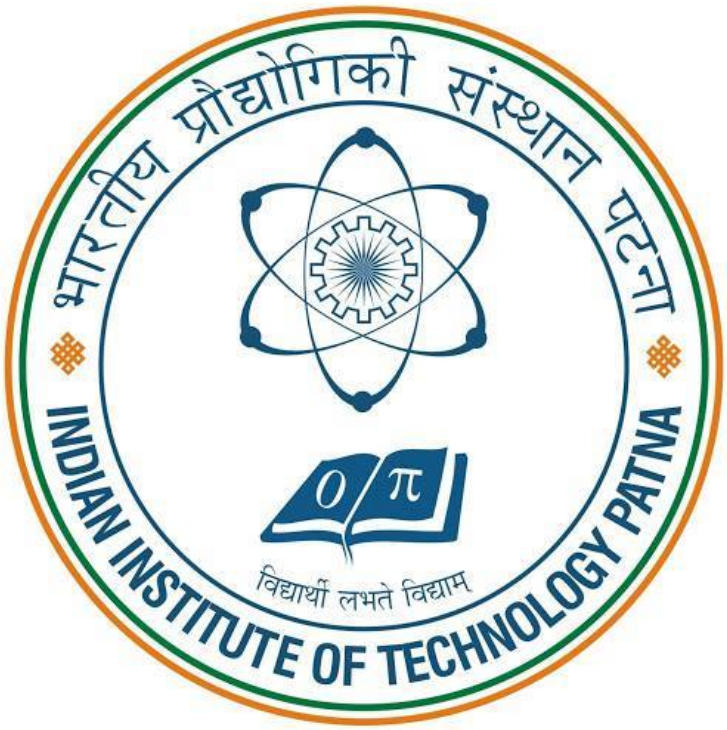
Last but not the least, we would like to thank our family and friends, for all that they meant to us during the crucial times of the completion of our internship.

Swadha Kumari (181029)

Karan Kumar (181040)

Priti Kumari (181038)

ABOUT THE ORGANISATION



Indian Institute of Technology Patna is one of the new IITs established by an Act of the Indian Parliament on August 06, 2008.

Patna which was known as Patliputra has been a centre of knowledge since long has been attracting visitors and scholars from many parts of the world such as China, Indonesia, Japan, Korea, Sri Lanka, among others. This has been a land of visionaries. Some of the legends from this region include Lord Gautam Buddha, Lord Mahavir, Guru Gobind Singh, the famous astronomer Aryabhatta and the first President of India, Dr. Rajendra Prasad.

IIT Patna has ten departments: These are Computer Science Engineering, Electrical Engineering, Mechanical Engineering, Chemical and Biochemical Engineering, Civil Environmental Engineering, Materials Science Engineering, Chemistry, Physics, Mathematics and Humanities Social Science departments.

IIT Patna campus is located at Bihta which is approximately 40 kms from Patna.

The Institute has developed modern facilities that are fully equipped with the state-of-the-art facilities (equipment software and machines) that are routinely used to train and educate students.

As of April 2016, IIT Patna has six hundred seventy students enrolled in its B Tech programs, one hundred sixty-one MTech students and two hundred fifty-nine PhD students. Since inception in 2008 and till April 2016, more than four hundred students have been awarded B.Tech. degree, more than seventy-five students have been awarded MTech. degree and more than 20 research scholars have been awarded PhD degree.

As of April 2016, faculty strength of IIT Patna is one hundred one (which includes one DST Ramanujan Faculty Fellow, one Adjunct Faculty, one DST INSPIRE Faculty and one Visiting

Professor). Faculty members are supported by more than seventy efficient staff (non-teaching) members. These numbers are anticipated to increase as the Institute is growing at a steady pace. The faculty members of IIT Patna have a wide range of academic and research experience. They have been trained in the top ranked Institutes within the country and abroad.

Research activity in IIT Patna has been published in high quality and peer-reviewed national and international journals. Please browse individual faculty member web-page for more information. Faculty members of IIT Patna have been also participating in national and international conferences of repute.

Contents

1. PROBLEM STATEMENT

Farmers have an increased prevalence of many acute and chronic health conditions including cardiovascular and respiratory disease, arthritis, skin cancer, hearing loss, and amputations. Other health outcomes have been little studies in the agricultural workplace, such as stress and adverse reproductive outcomes. [2]

Plant pathogens can be fungal, bacterial, viral or nematodes and can damage plant parts above or below the ground. Identifying symptoms and knowing when and how to effectively control diseases is an ongoing challenge for growers of cereals , pulses crops. Bacteria are single celled prokaryotic (no membrane around nucleus) microorganisms that are either free-living in soil or water or diseases of plants. As a disease of plants bacteria are a risk for primary producers as they impact upon market access and agricultural production. There are many types of viruses, viroid, prions and syndromes that have the potential to affect plant health. Viruses pose a serious risk for primary producers, as they can impact on market access and agricultural production. some of the world's major agricultural and livestock diseases.[3]

Good biosecurity measures on your property are vital for preventing the spread of plant diseases. Viruses can be spread by insect vectors. There are no pesticides that can be used to kill viruses, however they can be reduced and controlled by controlling these insect vectors with pesticides. Farmers know they lose crops to pests and plant diseases, but scientists have found that on a global scale, pathogens and pests are reducing crop yields for five major food crops by 10 percent to 40 percent, according to a report by a UC Agriculture and Natural Resources scientist and further members of the International Society for Plant Pathology. Wheat, rice, maize, soybean, and potato yields are reduced by pathogens and animal pests, including insects, scientists found in a global survey of crop health experts. Farmers know they lose crops to pests and plant diseases, but scientists have found that on a global scale, pathogens and pests are reducing crop yields for five major food crops by 10 percent to 40 percent, according to a report by a UC Agriculture and Natural Resources scientist and other members of the International Society for Plant Pathology. Wheat, rice, maize, soybean, and potato yields are reduced by pathogens and animal pests, including insects, scientists found in a global survey of crop health experts.[3]

Farmers know they lose crops to pests and plant diseases, but scientists have found that on a global scale, pathogens and pests are reducing crop yields for five major food crops by 10 percent to 40 percent, according to a report by a UC Agriculture and Natural Resources scientist and other members of the International Society for Plant Pathology. Wheat, rice, maize, soybean, and potato yields are reduced by pathogens and animal pests, including insects, scientists found in a global survey of crop health experts. Therefore, in the absence of assured and controlled water supply, the agricultural productivity in India is bound to be low.

The farmers in India have been adopting orthodox and inefficient method and technique of cultivation. It is only in recent years that the Indian farmers have started to adopt improved implements like steel ploughs, seed drills, barrows, hoes etc. to a limited extent only. Most of the farmers were relying on centuries old. Wooden plough and other implements. Such

adoption of traditional methods is responsible for low agricultural productivity in the country. Indian farmers are facing the problem of low income from their marketable surplus crops in the absence of proper organized markets and adequate transportation facilities. Scattered and sub-divided holdings are also creating serious problem for marketing their products.

Chronic bronchitis is more common among farmers than among the general population. The majority of farmers with this illness have a history of exposure to grain dust or work in swine confinement buildings. It is believed that cigarette smoking is additive and a cause of this illness. In addition, acute bronchitis has been described in grain farmers, especially during grain harvest. Proper rotation of crops is very much essential for successful agricultural operations as it helps to regain the fertility of the soil. As the farmers are mostly illiterate, they are not very much conscious about the benefit of crop rotation. Therefore, land loses its fertility to a considerable extent.[2]

2. RELATED WORK

Agriculture, farming or husbandry is a vital occupation since the history of mankind is maintained. The name agriculture represents all entities that came under the linear sequence of links of food chain for human beings. As humans are the smartest living species on this planet, so their smartness always provokes them to change and to innovate. This provoking has led to invention of wheel, advancements in living standards and styles, languages, life spending methodologies and countless more achievements. The restless attitude of mankind towards innovations has given birth to major inventions, which have not only provided the ease to human beings, but also improvements in efficiency which further lead to better productivity on the cost of less skills. The advancements in agriculture are necessary to balance the demand and supply as population is increasing day by day. As compare to the last fifty years and earlier, the demand of food has accelerated. To overcome the requirements, the deployment of modern technology over this vital source for humans is intolerable. With the use of modern and advanced technologies, efficiency of the agricultural industry can be improved, where it not only includes better productivity, but also lessens intra-field and inter-field losses present in conventional methods. From the beginning, agriculture is crucial part of human society due to the reality that man and agriculture are directly related to each other. This fact leads towards the advancement and enhancement of the typical, inappropriate and time consuming methodologies, used for agriculture. The fast moving world, new trends and technological advancement has changed the life style of people. Emerging new technologies are becoming an important part of routine. Smart homes and grid, smart cities smart campus, and smart farming are some of the whole advanced and upgraded, information and communication technologies that are helping humans to save time and get faster and acurate outcomes. [9]

2.1 TRENDS AND CONVENTIONS

Trends and research conventions are mainly focused on precise agriculture, database integration system and network information, virtual agriculture, expert systems, the connotation and extension, development stage and the impact of agricultural modernization for economic growth and improved life style of rural areas.Due to the direct impact on human life, agriculture is stepping towards modernization steadily. New trends are being introduced often, in order to meet the technological advancements. Comparing the past and continuing technological implementations it can be understood easily how most of the communication and information technologies are playing an effective role in modern agriculture. An increase in the demand for agriculture as a consequence of an abruptly growing population will enhance the need of efficient and actuate infrastructure support, in-order to fulfill the agricultural requirements of the modern society, without any interruption in its production. Wireless sensor networks and RFID tags are the most standardized technologies playing an active role in the smart agriculture. Many countries like

china, India, Korea, Brazil, Australia, several European countries and different American estates, are introducing agricultural technologies in order to strengthen their economy by using information and communication technologies for the improvement in agricultural and rural development. Hybrid architecture for localized agricultural information dissemination is the client server architecture, in conjunction with the mobile applications on smart-phones, which can be used to deliver the precise agricultural information to the farmers. Geographical data of mobile phones can further localize the required information needs of the farmers. The cattle and farm management, by using RFID tags and the recognition of cattle with the help of image processing, can lead to a decrease in the probability of viral spread.Considering these facts, the green houses are increasing in their popularity, every day. According to recent trends and technological development in Wireless Sensor Networks, it has been made possible to use WSN in the monitoring and control of the greenhouse parameter, in precision agriculture. Actuated sensor networks are being deployed for the management of green houses. Using wireless sensor networks will reduce the chances of human errors that can occur while investigating the facts, about the ideal method of irrigation suitable for all weather conditions, types of soils and different crop cultures. The usage of advanced technologies and automated machines, which is making the world soar to greater heights, experiences a lag when it comes to the farming either due to the lack of awareness or because of the unavailability of advanced facilities in the market, leading towards poverty in farming. In order to make the market more accessible to the farmers, the concept of e-farming is introduced. E-farming is the web application that will help the farmers to perform the agro-marketing leading to achieve success and increase in their standard of living. Smart agriculture is composed of many different technological implementations. These applications are replacing the tough, unreliable and time consuming traditional farming techniques with efficient, reliable and sustainable smart agriculture. Water irrigation context aware farming, pesticide control, remote monitoring, security control, environmental monitoring, precious agriculture, machine and process control, vehicle guidance, animal feeding facilities, traceability system, food packaging and inspection etc are a few examples.[9]

2.2 THE CONCEPT OF SMART

Generally speaking, if a machine/artifact or any system does something that we think an intelligent person can do, we consider the machine to be smart. Any system, process and domain is said to be smart if follows 6 different levels of intelligence.

• Adapting: The term adapting refers to the change to meet any particular requirements in terms of smart agriculture the changes would be referred to as environmental

• Sensing: The ability to sense the changes in surrounding or to observe any change.

• Inferring:It basically refers to conclusion which is based on results and observations.

• Learning:After getting conclusions and observed results the learning can be used to improve the methodologies used previously. It involves different type of information.

• Anticipating:It relates to thinking of something new and innovative which going to be happened or we can say it as the next level of anything.

2.2.1 SELF-ORGANIZING

: The processes referred to any intelligent system which has ability to sense and monitor and then change its parameters according to the need.Jotting all the 6 initials, smart agriculture can be composed of these main paradigms;

1. Smart Consumer
2. Smart Farmer
3. Smart farms

Smart consumer tends to the online access, for any end user to get information related to the productivity, particularly the consumer electronics in which one can be able to buy and sell the productions directly from the farm. This involves internet applications, web application, data base and online stores etc. The other end of smart consumer is the smart farmer side. It is the main node from where the farmer can directly interact with open market, without any extra expenses and involvement of third party. Any farm management system can be used to manage these outside activities. This node is then connected with the smart farm, which implies sensor nodes for humidity, moisture, weather, irrigation system, ware house management, cattle, pesticide detection and monitoring

2.3 RELEVANT TECHNOLOGIES

In the present era, there are various and constantly evolving technologies available. Many of them are suitable, in various locations and scenarios. These technologies are fairly equal to use and fine in their output regardless of their deployment in either rural or urban areas. While we are discussing agriculture and relevant technologies, there are two major groups to discuss; the sensors available and the communication platforms.

2.3.1 SENSORS

The Sensors available for remote deployments are not for a single measurement or coordinate collections. They are especially designed for gathering a bunch of information from concerned entity. The main composite sensors are available for climate, soil and plants.[9]

2.4 ISSUES AND PROPOSED SOLUTIONS

According to the International Assessment of Agricultural Science and Technology for Development (IAASTD).The widespread realization is that despite significant scientific and technological achievements in our ability to increase agricultural productivity, we have been less attentive to some of the unintended social and environmental consequences of our achievements. We are now in a good position to reflect on these consequences and to outline various policy options to meet the challenges ahead, perhaps best characterized as the need for food and livelihood security under increasingly constrained environmental conditions from within and outside the realm of agriculture and globalized economic systems.The synthesis proposed some of the factors like equitable environment and sustainability which can be used to improve the rural livelihoods and

can be useful in reducing hunger and poverty. In compliance with these factors, biological diversity and services for ecosystem rapid changes in climate and availability of water are some of the major concerns which are addressed. To fulfill the diverse needs of human life, there is a need for sustainability which requires the concern of the international collaboration.Apart from the technological advancement making its way towards smart agriculture, in order to benefit this industry which the humans directly associated with, along with its long lasting benefits also has some issues related to the agricultural advancement.

2.4.1 HUMAN BEHAVIOUR

The main barrier in the development of agriculture is the human behavior towards adapting new technology. It has always been hard for a common man to adopt something different from the traditional method. Most people from the ruler areas are associated with the farming and are not much educated and independent in technological advancement. This factor has widened the gap between the modern and ruler areas. It is evident that the electronic media can be a great means of reducing the hesitation of adapting new technologies by commercials and on-air campaigns.

2.4.2 AGRICULTURE

There are many issues directly associated with agriculture like grid, crop and soil monitoring, irrigation, pesticide and fertilization applications, and cattle farming. Information and communication technology can be a practical tool for overcoming these technical issues. Climate Changes The climatic change is biggest issue in agricultural paradigm, which directly affects each and every factor associated with farming. This natural conflict directly influences productivity and quality, leaving lasting and long-term impacts on food security. Quick solutions are needed for this issue. Pre-weather detection, temperature monitoring, climate changes, moisture levels, air flow and pressure, rain and extreme weather prediction are few of the many solutions.

2.4.3 MARKET/VENDORS

Due to the long geographical distances between the open market and the actual farm the field cost of the production is increased and will directly benefit the brokers or the third person involved, leaving the farmers inadequately paid for their efforts.When it comes to venders of agriculture equipment's equal opportunities should be given to the manufacturing industries. Monopoly should be discouraged. This will improve the quality and cost reduction. Online services and applications can be used to provide direct interaction between farmers and consumers; e-store can be a better option. Standardization in equipment manufacturing can improve the productivity and compatibility between different vendors.

2.4.4 REGULATIONS

Reforms are needed on national and international level. Government policies and regulations are needed. The formation of regulating authority and its up gradation for technologically advanced agriculture is needed in order to provide an accessible and open market.[9]

3. INTRODUCTION TO THE PROPOSED MODEL

The system herein proposes a decentralised application via which a farmer could easily apply for an insurance of his crops, based on the crop type, namely, the rabi and kharif crops, the area of farm and the location of the place. Here, the farmer will have to submit some of the images of his crops. A machine learning model at the backend would go through these images and if the crop are not spoilt from before, the insurance would be approved and the farmer would be allotted a unique policy identification number for that specific crop, and he would have to pay a certain premium amount, which specifically depends on the crop.

Now, at a later stage, if the farmer finds that his crops have got affected with a certain disease and he does not know the severity of the disease and up to what extent this disease would cost him, he could simply approach this application. Here, he will just have to upload the images of the affected portions and within some seconds, the application would tell him about the disease that the crop is subjected to. Apart from this, the farmer would be offered a coverage amount based on the severity of the disease, like if the crops could no longer be cured, then the full coverage amount would be offered, however, if they could be cured, then a certain percentage of the coverage amount would be given to the farmers.

Moreover, if there are farmers who have not taken any kind of insurance for their crops, and their crops are on the verge of getting wasted, they could also approach this application to get a detailed information of the disease, as this system is trained with the machine learning models which have a high accuracy and are highly precisive. This is highly beneficial in situations when the farmers could not study their crops with their naked eye observation and miss out the small signs, however, this system could study out the smallest details on the leaves and would detect even the diseases that are at a premature stage.

The system herein is designed to be a decentralised application, as here there is no intervention of third-party mediators but the information of the policy and the crops of the farmers remains on the blockchain ledger which could not be tampered with. This hereby builds the trust among the farmers. The backend code here runs on the decentralised peer-to-peer networks, as opposed to the typical applications where the backend code is running on centralised servers. This proves to be hassle free in comparison to the traditional methods and the age-old conventions of getting an insurance policy and getting them approved in cases of wastage. This has incorporated a single window system for the farmers, by providing all the services via a single application just through some clicks.

4. TECHNOLOGY USED

4.1 BLOCKCHAIN

A blockchain is a constantly growing ledger which keeps a permanent record of all the transactions that have taken place in a secure, chronological, and immutable way. A blockchain is a chain of blocks which contain information. Each block records all of the recent transactions, and once completed goes into the blockchain as a permanent database. Each time a block gets completed, a new block is generated. Blockchain technology can be integrated into multiple areas. The primary use of blockchains is as a distributed ledger for cryptocurrencies. It shows great promise across a wide range of business applications like Banking, Finance, Government, Healthcare, Insurance, Media.[8]

It is a system of recording information in a way that makes it difficult or impossible to change, hack, or cheat the system. It is a system of recording information in a way that makes it difficult or impossible to change, hack, or cheat the system. Bitcoin was designed to solve this problem by using a specific type of database called a blockchain. Most normal databases, such as an SQL database, have someone in charge who can change the entries (e.g. giving themselves a million X dollars). Blockchain is different because nobody is in charge; it's run by the people who use it. What's more, bitcoins can't be faked, hacked or double spent – so people that own this money can trust that it has some value.

In the financial industry, blockchain can allow the quicker settlement of trades. It does not take a lengthy process for verification, settlement, and clearance. It is because of a single version of agreed-upon data available between all stakeholders. blockchain register transactions in a chronological order which certifies the inalterability of all operations, means when a new block is added to the chain of ledgers, it cannot be removed or modified. Blockchain certifies and verifies the identities of each interested parties. This removes double records, reducing rates and accelerates transactions.[8]

Blockchain uses very advanced cryptography to make sure that the information is locked inside the blockchain. It uses Distributed Ledger Technology where each party holds a copy of the original chain, so the system remains operative, even the large number of other nodes fall. It allows each party to transact directly with each other without requiring a third-party intermediary.

It is decentralized because there is no central authority supervising anything. There are standards rules on how every node exchanges the blockchain information. This method ensures that all transactions are validated, and all valid transactions are added one by one. Blockchain Cryptocurrency is a type of digital asset which is used to exchange value between parties. It uses strong cryptography to secure financial transactions and control the creation of new units of that currency and verify the transfer of assets.

Cryptocurrency does not exist physically. We know that the government prints the government

currencies like fiat currency such as Dollar, Yen or Yuan itself. It means there is a centralized institution exists which can create thousands or millions or billions more of that currency. Unlike government currencies like bitcoin, these type of currencies is created by the same mathematical formulas that make the cryptocurrency work. Thus, cryptocurrencies use decentralized control, which works through distributed ledger technology that serves as a public financial transaction database.[6]

Bitcoin is generally known as the first decentralized cryptocurrencies. It is created by the network of thousands of specific nodes called miners. The miners can process the bitcoin transactions in the blockchain network. Since the release of bitcoin, more than 4000 alternative variants of bitcoin are available now. In the blockchain technology, bitcoin is the best-known implementation of the blockchain. There is a lot of development and the direction is based on the premise of what blockchain does to enable Bitcoin to happen. We can learn and expand how it can spread into so many different areas.[6]

A blockchain distributed ledger is a type of database that is consensually shared, replicated, and synchronized among the members of a decentralized network. All the information on this ledger is securely and accurately stored using cryptography. This information can be accessed by using keys and cryptographic signatures. The distributed ledger allows transactions to have public witnesses, which makes cyberattack more difficult. It records the transactions such as the exchange of assets or data, among the participants in the network.[1]

Blockchain technology has enormous potential in creating trustless, decentralized applications. But it is not perfect. There are certain barriers which make the blockchain technology not the right choice and unusable for mainstream application. We can see the limitations of blockchain technology in the following image. There is a lot of discussion about blockchain, but people do not know the true value of blockchain and how they could implement it in different situations.

Today, there are a lot of developers available who can do a lot of different things in every field. But in the blockchain technology, there are not so many developers available who have specialized expertise in blockchain technology. Hence, the lack of developers is a hindrance to developing anything on the blockchain. In immutable, we cannot make any modifications to any of the records. It is very helpful if you want to keep the integrity of a record and make sure that nobody ever tampers with it. But immutability also has a drawback. We can understand this, in the case, when you want to make any revisions, or want to go back and make any reversals. For example, you have processed payment and need to go back and make an amendment to change that payment. As we know, blockchain is built on cryptography, which implies that there are different keys, such as public keys and private keys. When you are dealing with a private key, then you are also running the risk that somebody may lose access to your private key. It happens a lot in the early days when bitcoin wasn't worth that much. People would just collect a lot of bitcoin, and then suddenly forgot what the key was, and those may be worth millions of dollars today.[1]

Blockchain like bitcoin has consensus mechanisms which require every participating node to verify the transaction. It limits the number of transactions a blockchain network can process. So bitcoin was not developed to do the large scale volumes of transactions that many of the other institutions are doing. Currently, bitcoin can process a maximum of seven transactions per second. In the blockchain, we know that a block can be created in every 10 minutes. It is because every transaction made must ensure that every block in the blockchain network must reach a common consensus. Depending on the network size and the number of blocks or nodes involved

in a blockchain, the back-and-forth communications involved to attain a consensus can consume a considerable amount of time and resources. Proof of Work(PoW) is the original consensus algorithm in a blockchain network. The algorithm is used to confirm the transaction and creates a new block to the chain. In this algorithm, minors (a group of people) compete against each other to complete the transaction on the network. The process of competing against each other is called mining. As soon as miners successfully created a valid block, he gets rewarded. The most famous application of Proof of Work(PoW) is Bitcoin.

Producing proof of work can be a random process with low probability. In this, a lot of trial and error is required before a valid proof of work is generated. The main working principle of proof of work is a mathematical puzzle which can easily prove the solution. Proof of work can be implemented in a blockchain by the Hashcash proof of work system.[1]

4.2 ETHEREUM

Ethereum is the second-largest cryptocurrency platform by market capitalization, behind bitcoin. It is a decentralized open source blockchain featuring smart contract functionality. Ether is the cryptocurrency generated by Ethereum miners as a reward for computations performed to secure the blockchain. Ethereum serves as the platform for over 260,000 different cryptocurrencies, including 47 of the top 100 cryptocurrencies by market capitalization.

Ethereum provides a decentralized virtual machines, the Ethereum Virtual Machine (EVM), which can execute scripts using an international network of public nodes./The virtual machine's instruction set, in contrast to others like bitcoin script, is turing -complete. "Gas", an internal transaction pricing mechanism, is used to mitigate spam and allocate resources on the network. [11]

Ethereum is open access to digital money and data-friendly services for everyone – no matter your background or location. It's a community-built technology behind the cryptocurrency Ether (ETH) and thousands of applications you can use today. Ethereum is a technology that lets you send cryptocurrency to anyone for a small fee. It also powers applications that everyone can use and no one can take down.

Ethereum is currently developing and planning to implement a series of upgrades called Ethereum 2.0. Current specifications for Ethereum 2.0 include a transition the proof of stake and an increase in transaction throughput using technology. Ethereum was initially described in a white paper by a programmer and co-founder in late 2013 with a goal of building decentralized applications. Buterin had argued that Bitcoin needed a scripted language for application development. Failing to gain agreement, he proposed the development of a new platform with a more general scripting language.

Ethereum was announced at the North American Bitcoin Conference in Miami, in January 2014. Several codenamed prototypes of the Ethereum platform were developed by the Ethereum Foundation, as part of their Proof-of-Concept series, prior to the official launch of the Frontier network. "Olympic" was the last of these prototypes, and public beta pre-release. The Olympic network provided users with a bug bounty of 25,000 Ether for stress testing the limits of the Ethereum blockchain. As with other cryptocurrencies, the validity of each Ether is provided by a blockchain, which is a continuously growing list of records, called blocks, which are linked and secured using cryptocurrency.[11]

By design, the blockchain is inherently resistant to modification of the data. It is an open, distributed ledger that records transactions between two parties efficiently and in a verifiable and permanent way. Unlike Bitcoin, Ethereum operates using accounts and balances in a manner called state transitions. This does not rely upon unspent transaction output (UTXOs). The state denotes the current balances of all accounts and extra data. The state is not stored on the blockchain, it is stored in a separate merkle Patricia tree. [11]

A cryptocurrency wallet stores the public and private keys or "addresses" which can be used to receive or spend ether. These can be generated through BIP 39 style mnemonics for a BIP 32 "HD Wallet". In Ethereum, this is unnecessary as it does not operate in a UTXO scheme. With the private key, it is possible to write in the blockchain, effectively making an Ether transaction.To send the Ethereum value token Ether to an account, you need the hash of the public key of that account. Ethereum accounts are pseudonymous in that they are not linked to individual persons, but rather to one or more specific addresses.

Ether is a fundamental token for operation of Ethereum, which thereby provides a public distributed ledger for transactions. It is used to pay for gas, a unit of computation used in transactions and other state transitions. Mistakenly, this currency is also referred to as Ethereum.It is listed under the ticker symbol ETH and traded on cryptocurrency exchanges, and the Greek uppercase Xi character () is generally used for its currency symbol. It is also used to pay for transaction fees and computational services on the Ethereum network. [5]

Ethereum's smart contracts are based on different computer languages, which developers use to program their own functionalities. Smart contracts are high-level programming abstractions that are compiled down to EVM bytecode and deployed to the Ethereum blockchain for execution. The Ethereum Virtual Machine (EVM) is the runtime environment for smart contracts in Ethereum. It is a 256-bit register stack, designed to run the same code exactly as intended. It is the fundamental consensus mechanism for Ethereum. The formal definition of the EVM is specified in the Ethereum Yellow Paper. In Ethereum all smart contracts are stored publicly on every node of the blockchain, which has costs.[5]

Being a blockchain means it is secure by design and is an example of a distributed computing system with high Byzantine fault tolerance. The downside is that performance issues arise in that every node is calculating all the smart contracts in real time, resulting in lower speeds.] As of January 2016, the Ethereum protocol could process about 25 transactions per second. In comparison, the Visa payment platform processes 45,000 payments per second leading some to question the scalability of Ethereum. Ethereum-based customized software and networks, independent from the public Ethereum chain, are being tested by enterprise software companies. [11]

4.3 INTERPLANETARY FILE SYSTEM (IPFS)

IPFS (Interplanetary File System) is a peer to peer, version controlled, content-addressed file system. It makes use of Computer Science concepts like Distributed Hash Table, BitSwap (Inspired by BitTorrent), MerkleDag (Inspired by The Git Protocol). There are multiple applications currently being built on top of IPFS. The current default way to exchange data across the Internet is HTTP, but it fails in some cases. Large files cannot be transferred using HTTP, data is not permanent on HTTP, HTTP mainly uses a Client-Server protocol which leads to low latency

and makes it difficult to establish a peer to peer connection, also real-time media streaming is difficult on HTTP. All of these failures are overcome using IPFS.

Unlike HTTP which is IP addressed, an IPFS network is content addressed. Which means, when any data is uploaded on an IPFS network, it returns a Hash and the data is then requested using that hash. Anyone can provide storage on the IPFS network and everyone is incentivized with crypto tokens. Data is distributed and replicated throughout the network which leads to data permanence. While requesting data it searches for the nearest copy of that data which leads to high latency and overcomes any bottleneck points.

As the data is completely distributed, it has no scope for centralization of data. IPFS could be seen as a single BitTorrent swarm, exchanging objects within one Git repository. The current default way to exchange data across the Internet is HTTP, but it fails in some cases. Large files cannot be transferred using HTTP, data is not permanent on HTTP, HTTP mainly uses a Client-Server protocol which leads to low latency and makes it difficult to establish a peer to peer connection, also real-time media streaming is difficult on HTTP. All of these failures are overcome using IPFS.

IPFS could be seen as a single BitTorrent swarm, exchanging objects within one Git repository. Distributed hash table is used to store and retrieve data across nodes in the network. It is a class which is similar to hash tables. Using a DHT, any node on the network can request the value corresponding to a hash key.

Blockchain exchange is used in the BitTorrent Protocol (also known as BitSwap) to exchange data between nodes. It is a peer to peer file sharing protocol which coordinates data exchange between untrusted swarms .It uses a tit for tat strategy which rewards nodes that contribute to each other and punishes nodes that only request resources. This helps an IPFS node in retrieving multiple parts of data parallelly.

It uses Merkle DAG similar to the one used in the Git Version Control system. It is used to track change to files on the network in a distributed friendly way. Data is content-addressed, by the cryptographic hash of the content. Every node on the network is identified using a NodeID which is nothing but the hash of its public key. Everyone on the network can store files on their local. storage and they are incentivized to do so. Each node maintains a DHT which is used to find out Ids of other peers on the network and what data those peers can serve.

Users in a local network can communicate with each other, even if the Wide Area network is blocked for some reason. Since no servers are required, creators can distribute their work without any cost. Data loads faster as it has higher bandwidth. IPFS installation has a lot of hasseles, it is not at all user friendly. IPFS consumes lot of bandwidth which is not appreciated by metered internet users. IPFS currently is used by tech enthusiast and normal people don't tend to set up their own node, which leads to shortage of nodes on the network. IPFS synthesizes various best systems and protocols to date. IPFS is an ambitious vision of a new decentralized Internet infrastructure, upon which many different kinds of applications can be built in the future.

The list of problems go on and it is no surprise that a technology more than 20 years old is becoming more noticeably outdated in an age of technological innovation. IPFS provides the distributed storage and file system that the Internet needs to achieve its true potential. Instead of downloading files from single servers, in IPFS, you ask peers in the network to give you a path to a file rather than it coming from a central server. This enables high volume data distribution with high efficiency, historic versioning, resilient networks, and persistent availability of content secured and verified through cryptographic hashing and distributed across a network of peers.

The design of the protocol provides historic versioning of the Internet like with Git. Each file and all blocks within it are given a unique identifier, which is a cryptographic hash. Duplicates are removed across the network and version history is tracked for every file.

IPFS links file structures to each other using Merkle links and every file can be found by human-readable names using a decentralized naming system called IPNS. Content has a unique identifier that is the cryptographic hash of the file. Content has a unique identifier that is the cryptographic hash of the file. Data is verified with its checksum, so if the hash changes, then IPFS will know the data is tampered with. Integration of IPFS with blockchain technology seems to be a perfect fit. [7]

5. DEPENDENCIES

5.1 TRUFFLE

Truffle is a framework for building, testing, and deploying applications on the Ethereum network that was founded by Tim Coulter. The Truffle Framework consists of three primary development frameworks for Ethereum smart contract and decentralized application (dApp) development called Truffle, Ganache, and Drizzle.

npm install -g truffle

5.1.1 REQUIREMENTS

- NodeJS v8.9.4 or later
- Windows, Linux or Mac OS X

Truffle also requires that one have a running Ethereum client which supports the standard JSON RPC API (which is nearly all of them). There are many to choose from, and some better than others for development. We'll discuss them in detail in the Choosing an Ethereum client section

To use most Truffle commands, one need to run them against an existing Truffle project. So the first step is to create a Truffle project.

One can use the 'truffle unbox ' command to download any of the Truffle Boxes. For example: truffle unbox metacoin

Once this operation is completed, one will have a project structure with the following items:

- contracts/: Directory for Solidity contracts
- migrations/: Directory for scriptable deployment files
- test/: Directory for test files for testing your application and contracts
- truffle-config.js: Truffle configuration file

All of the contracts are located in your project's contracts/ directory. As contracts are written in Solidity, all files containing contracts will have a file extension of .sol. Associated Solidity libraries will also have a .sol extension. With a bare Truffle project (created through truffle init), you're given a single Migrations.sol file that helps in the deployment process. If you're using a Truffle Box, you will have multiple files here. To compile a Truffle project, change to the root of the directory where the project is located and Upon first run, all contracts will be compiled.

Upon subsequent runs, Truffle will compile only the contracts that have been changed since the last compile. If one like to override this behaviour, run the above command with the –all option. Artifacts of your compilation will be placed in the build/contracts/ directory, relative to your project root. (This directory will be created if it does not exist.) These artifacts are integral to the inner workings of Truffle, and they play an important part in the successful deployment of your application. You should not edit these files as they'll be overwritten by contract compilation and deployment.Truffle supports dependencies installed via both EthPM and NPM. To import contracts from a dependency, use the following syntaximport "somepackage/SomeContract.sol";

5.2 GANACHE

Ganache is a personal blockchain for rapid Ethereum and Corda distributed application development. You can use Ganache across the entire development cycle; enabling you to develop, deploy, and test your dApps in a safe and deterministic environment.

Ganache UI is desktop application supporting both Ethereum and Corda technology. In addition, an Ethereum version of ganache is available as a command-line tool: ganache-cli (formerly known as the TestRPC).

1. Install Ganache

Next, double-click on the downloaded file, follow the prompts, and you're up and running.

2. Create a Workspace

When you open Ganache for the first time, you'll see the home screen. On this screen you're prompted to load an existing workspace (if any exist), create a new custom workspace, or quickstart a one-click blockchain with default options. For now, let's go with a quickstart workspace. Select the desired blockchain from the QUICKSTART drop down; you can choose to start an Ethereum node or Corda network, then click the QUICKSTART button.

5.2.1 LINKING A TRUFFLE PROJECT

To link a project, enter the settings by clicking the gear icon in the upper right.

One should be seeing the WORKSPACE settings pane; if not, you can get there by clicking the WORKSPACE tab in the top left.

From here, there is a section labelled TRUFFLE PROJECTS. Beneath this box, click the button ADD PROJECT. A file selection popup will appear. Navigate to the folder of your Truffle project, and select the truffle-config.js or truffle.js configuration file. The file you pick must be either named truffle-config.js or truffle.js for Ganache to correctly load it. After selecting the file, one will see it listed in the TRUFFLE PROJECTS section.

You can add multiple projects to a workspace. After you're finished adding projects you can click the SAVE AND RESTART (SAVE WORKSPACE if this is a new workspace) button in the top right. After at least one workspace has been created, the home screen will now have a list of workspaces for you to choose from. You can scroll through the list to find the desired workspace, and then load the workspace by clicking its name.

5.3 REACT

React (also known as React.js or ReactJS) is an open-source JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with rendering data to the DOM, and so creating React applications usually requires the use of additional libraries for state management and routing. Redux and React Router are respective examples of such libraries.

5.3.1 REACT TRUFFLE BOX

This box comes with everything you need to start using smart contracts from a react app. This is as barebones as it gets, so nothing stands in your way.

React has been designed from the start for gradual adoption, and you can use as little or as much React as you need. Whether you want to get a taste of React, add some interactivity to a simple HTML page, or start a complex React-powered app, the links in this section will help you get started.

5.3.2 ONLINE PLAYGROUNDS

If you're interested in playing around with React, you can use an online code playground. Try a Hello World template on CodePen, CodeSandbox, Glitch, or Stackblitz.

If you prefer to use your own text editor, you can also download this HTML file, edit it, and open it from the local filesystem in your browser. It does a slow runtime code transformation, so we'd only recommend using this for simple demos.

5.3.3 ADD REACT TO A WEBSITE

You can add React to an HTML page in one minute. You can then either gradually expand its presence, or keep it contained to a few dynamic widgets.

5.3.4 CREATE A NEW REACT APP

When starting a React project, a simple HTML page with script tags might still be the best option. It only takes a minute to set up!

As your application grows, you might want to consider a more integrated setup. There are several JavaScript toolchains we recommend for larger applications. Each of them can work with little to no configuration and lets you take full advantage of the rich React ecosystem. Learn how.

An element describes what you want to see on the screen:

const element = <h1>Hello, world</h1>;

Unlike browser DOM elements, React elements are plain objects, and are cheap to create. React DOM takes care of updating the DOM to match the React elements.

5.3.5 COMPONENTS

React code is made of entities called components. Components can be rendered to a particular element in the DOM using the React DOM library. When rendering a component, one can pass in values that are known as "props".

ReactDOM.render(<Greeter greeting="Hello World!" />,

document.getElementById('myReactApp'));

The two primary ways of declaring components in React is via functional components and class-based components.

Parallel native technology for creating reusable building blocks of the web — Web Components. Advantage over React components — ability to create components not only for React but also for Angular, other libraries/frameworks, and for projects without any external dependency.

5.3.6 FUNCTIONAL COMPONENTS

Functional components are declared with a function that then returns some JSX.

const Greeting = (props) => Hello, props.name! ;

5.3.7 CLASS BASED COMPONENTS

Class-based components are declared using ES6 classes.

class ParentComponent extends React.Component

state = color: 'green' ;

render()

return (

<ChildComponent color=this.state.color />

);

5.3.8 VIRTUAL DOM

Another notable feature is the use of a virtual Document Object Model, or virtual DOM. React creates an in-memory data-structure cache, computes the resulting differences, and then updates the browser's displayed DOM efficiently. This process is called reconciliation. This allows the programmer to write code as if the entire page is rendered on each change, while the React libraries only render subcomponents that actually change. This selective rendering provides a major performance boost. It saves the effort of recalculating the CSS style, layout for the page and rendering for the entire page.

5.3.9 LIFECYCLE METHODS

Lifecycle methods use a form of hooking that allows execution of code at set points during a component's lifetime.

- shouldComponentUpdate allows the developer to prevent unnecessary re-rendering of a component by returning false if a render is not required.
- componentDidMount is called once the component has "mounted" (the component has been created in the user interface, often by associating it with a DOM node). This is commonly used to trigger data loading from a remote source via an API.
- componentWillUnmount is called immediately before the component is torn down or "unmounted". This is commonly used to clear resource demanding dependencies to the component that will not simply be removed with the unmounting of the component (e.g., removing any setInterval() instances that are related to the component, or an "eventListener" set on the "document" because of the presence of the component)
- render is the most important lifecycle method and the only required one in any component. It is usually called every time the component's state is updated, which should be reflected in the user interface.

5.3.10 JSX

JSX, or JavaScript XML , is an extension to the JavaScript language syntax. Similar in appearance to HTML, JSX provides a way to structure component rendering using syntax familiar to many developers. React components are typically written using JSX, although they do not have to be (components may also be written in pure JavaScript). JSX is similar to another extension syntax created by Facebook for PHP called XHP.

React does not attempt to provide a complete "application library". It is designed specifically for building user interfaces and therefore does not include many of the tools some developers might consider necessary to build an application. This allows the choice of whichever libraries the developer prefers to accomplish tasks such as performing network access or local data storage. Common patterns of usage have emerged as the library matures.

5.3.11 USE OF THE FLUX ARCHITECTURE

To support React's concept of unidirectional data flow (which might be contrasted with AngularJS's bidirectional flow), the Flux architecture represents an alternative to the popular model-view-controller architecture. Flux features actions which are sent through a central dispatcher to a store, and changes to the store are propagated back to the view. When used with React, this propagation is accomplished through component properties.

Flux can be considered a variant of the observer pattern. A React component under the Flux architecture should not directly modify any props passed to it, but should be passed callback functions that create actions which are sent by the dispatcher to modify the store. The action is an object whose responsibility is to describe what has taken place: for example, an action describing one user "following" another might contain a user id, a target user id, and the type USERFOLLOWEDANOTHERUSER.

The stores, which can be thought of as models, can alter themselves in response to actions received from the dispatcher.

This pattern is sometimes expressed as "properties flow down, actions flow up". Many implementations of Flux have been created since its inception, perhaps the most well-known being Redux, which features a single store, often called a single source of truth. [4]

5.4 WEB3

Web 3 refers to an Internet that is made possible by decentralized networks, such as Bitcoin and Ethereum. The key innovation of these networks is the creation of platforms that no single entity controls, yet everyone can still trust. That's because every user and operator of these networks must follow the same set of hard-coded rules, known as consensus protocols.

The secondary innovation is that these networks allow value or money to be transferred between accounts. These two things—decentralization and Internet money—are the keys to understanding Web 3.

With Web 3, the network is decentralized, so no one entity controls it, and the decentralized applications (dapps) that are built on top of the network are open. The openness of the decentralized web means that no single party can control data or limit access. Anyone is able to build and connect with different dapps without permission from a central company.

On Web 3, money is native. Instead of having to rely on the traditional financial networks that are tied to governments and restricted by borders, money on Web 3 is instant, global, and permissionless. It also means tokens and cryptocurrencies can be used to design completely new business models and economies, a field increasingly becoming known as tokenomics.

For example, advertising on the decentralized web would not need to rely on selling users' data to advertisers, but could instead reward users with a token for viewing ads.

5.4.1 GETTING STARTED

The web3.js library is a collection of modules that contain functionality for the ethereum ecosystem.

web3-eth is for the ethereum blockchain and smart contracts.

web3-shh is for the whisper protocol, to communicate p2p and broadcast.

web3-bzz is for the swarm protocol, the decentralized file storage.

web3-utils contains useful helper functions for Dapp developers.

5.4.2 ADDING WEB3.JS

First you need to get web3.js into your project. This can be done using the following methods:

npm: npm install web3

yarn: yarn add web3

pure js: link the dist/web3.min.js

After that you need to create a web3 instance and set a provider.

Most Ethereum-supported browsers like MetaMask have an EIP-1193 compliant provider available at window.ethereum.

For web3.js, check Web3.givenProvider.

If this property is null you should connect to a remote/local node.

// In Node.js use: const Web3 = require('web3');

let web3 = new Web3(Web3.givenProvider || "ws://localhost:8545"); That's it! now you can use the web3 object.

5.4.3 CALLBACKS PROMISES EVENTS

To help web3 integrate into all kinds of projects with different standards we provide multiple ways to act on asynchronous functions.

Most web3.js objects allow a callback as the last parameter, as well as returning promises to chain functions.

Ethereum as a blockchain has different levels of finality and therefore needs to return multiple "stages" of an action. To cope with requirement we return a "promiEvent" for functions like web3.eth.sendTransaction or contract methods. This "promiEvent" is a promise combined with an event emitter to allow acting on different stages of action on the blockchain, like a transaction.

PromiEvents work like a normal promises with added on, once and off functions. This way developers can watch for additional events like on "receipt" or "transactionHash".

web3.eth.sendTransaction(from: '0x123...', data: '0x432...')

.once('sending', function(payload) ...)

.once('sent', function(payload) ...)

.once('transactionHash', function(hash) ...)

.once('receipt', function(receipt) ...)

.on('confirmation', function(confNumber, receipt, latestBlockHash) ...)

.on('error', function(error) ...)

.then(function(receipt) // will be fired once the receipt is mined);

5.4.4 JSON INTERFACE

The json interface is a json object describing the Application Binary Interface (ABI) for an Ethereum smart contract.

Using this json interface web3.js is able to create JavaScript object representing the smart contract and its methods and events using the web3.eth.Contract object.

5.4.5 SPECIFICATION

5.4.5.1 FUNCTIONS

- type: "function", "constructor" (can be omitted, defaulting to "function"; "fallback" also possible but not relevant in web3.js);
- name: the name of the function (only present for function types);

- constant: true if function is specified to not modify the blockchain state;
- payable: true if function accepts ether, defaults to false;
- stateMutability: a string with one of the following values:
  - pure (specified to not read blockchain state)
  - view (same as constant above)
  - nonpayable and payable (same as payable above)
- inputs: an array of objects, each of which contains:
  - name: the name of the parameter
  - type: the canonical type of the parameter
- outputs: an array of objects same as inputs, can be omitted if no outputs exist.

5.4.5.2 EVENTS

- type: always "event"
- name: the name of the event;
- inputs: an array of objects, each of which contains:
  - name: the name of the parameter;
  - type: the canonical type of the parameter.
  - indexed: true if the field is part of the log's topics, false if it one of the log's data segment.

5.5 METAMASK PLUG-IN

MetaMask is a very useful tool that plays a pivotal role in allowing you to make your foray into the world of blockchain. First and foremost, MetaMask is an Ethereum wallet. It allows one to

• Create accounts for use in the various Ethereum networks

• It maintains the private keys for accounts so that one can export them or import new accounts.

• Switch between the various Ethereum networks, so that accounts can reflect the correct balance for each network

• Perform transactions between accounts

• One can transfer Ethers from one account to another. One can also hold tokens in your MetaMask accounts.

• One can also view your detail transactions on Etherscan, a blockchain explorer.

MetaMask is an extension for accessing Ethereum enabled distributed applications, or "Dapps" in your browser!

The extension injects the Ethereum web3 API into every website's javascript context, so that dapps can read from the blockchain.

MetaMask also lets the user create and manage their own identities (via private keys, local client wallet and hardware wallets like Trezor^{TM}), so when a Dapp wants to perform a transaction and write to the blockchain, the user gets a secure interface to review the transaction, before approving or rejecting it.

Because it adds functionality to the normal browser context, MetaMask requires the permission to read and write to any webpage. You can always "view the source" of MetaMask the way you do any Chrome extension.

Enables access to:

- Web 3.0
- Dapps
- NFTs
- erc20
- tokens
- ICOs
- erc271

5.5.1 INSTALLING METAMASK

The easiest way to install MetaMask is to use the Chrome browser and install MetaMask as a Chrome extension.

To install the MetaMask extension:

• Launch Chrome and navigate to the Chrome Web Store.

• Search for MetaMask.

• You should now be able to see the MetaMask extension in the search result. Click Add to Chrome (see Figure 5-2). You will be prompted to add MetaMask to Chrome. Click Add extension

Once MetaMask is added to Chrome, you will be able to see its icon appear on the top right corner of the browser

5.6 NODE-JS

Node.js is an open-source, cross-platform, JavaScript runtime environment (Framework) that executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language, rather than different languages for server- and client-side scripts.

Though .js is the standard filename extension for JavaScript code, the name "Node.js" doesn't refer to a particular file in this context and is merely the name of the product. Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize

throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games).

The Node.js distributed development project was previously governed by the Node.js Foundation, and has now merged with the JS Foundation to form the OpenJS Foundation, which is facilitated by the Linux Foundation's Collaborative Projects program.

Corporate users of Node.js software include GoDaddy, Groupon, IBM, LinkedIn, Microsoft, Netflix, PayPal, Rakuten, SAP, Voxer, Walmart, and Yahoo!.

Node.js allows the creation of Web servers and networking tools using JavaScript and a collection of "modules" that handle various core functionalities. Modules are provided for file system I/O, networking (DNS, HTTP, TCP, TLS/SSL, or UDP), binary data (buffers), cryptography functions, data streams, and other core functions. Node.js's modules use an API designed to reduce the complexity of writing server applications.

JavaScript is the only language that Node.js supports natively, but many compile-to-JS languages are available. As a result, Node.js applications can be written in CoffeeScript, Dart, TypeScript, ClojureScript and others.

Node.js is primarily used to build network programs such as Web servers. The most significant difference between Node.js and PHP is that most functions in PHP block until completion (commands only execute after previous commands finish), while Node.js functions are non-blocking (commands execute concurrently or even in parallel, and use callbacks to signal completion or failure).

Node.js is officially supported on Linux, macOS and Microsoft Windows 8.1 and Server 2012 (and later), with tier 2 support for SmartOS and IBM AIX and experimental support for FreeBSD. OpenBSD also works, and LTS versions available for IBM i (AS/400). The provided source code may also be built on similar operating systems to those officially supported or be modified by third parties to support others such as NonStop OS and Unix servers.

5.6.1 PLATFORM ARCHITECTURE

Node.js brings event-driven programming to web servers, enabling development of fast web servers in JavaScript. Developers can create scalable servers without using threading, by using a simplified model of event-driven programming that uses callbacks to signal the completion of a task. Node.js connects the ease of a scripting language (JavaScript) with the power of Unix network programming.

Node.js was built on the Google V8 JavaScript engine since it was open-sourced under the BSD license. It is proficient with internet fundamentals such as HTTP, DNS, TCP. JavaScript was also a well-known language, making Node.js accessible to the web development community.

5.6.2 TECHNICAL DETAILS

Node.js is a JavaScript runtime environment that processes incoming requests in a loop, called the event loop.

5.6.2.1 THREADING

Node.js operates on a single-thread event loop, using non-blocking I/O calls, allowing it to support tens of thousands of concurrent connections without incurring the cost of thread context switching. The design of sharing a single thread among all the requests that use the observer pattern is intended for building highly concurrent applications, where any function performing I/O must use a callback. To accommodate the single-threaded event loop, Node.js uses the libuv library—which, in turn, uses a fixed-sized thread pool that handles some of the non-blocking asynchronous I/O operations.

A thread pool handles the execution of parallel tasks in Node.js. The main thread function call posts tasks to the shared task queue, which threads in the thread pool pull and execute. Inherently non-blocking system functions such as networking translate to kernel-side non-blocking sockets, while inherently blocking system functions such as file I/O run in a blocking way on their own threads. When a thread in the thread pool completes a task, it informs the main thread of this, which in turn, wakes up and executes the registered callback.

A downside of this single-threaded approach is that Node.js doesn't allow vertical scaling by increasing the number of CPU cores of the machine it is running on without using an additional module, such as cluster, StrongLoop Process Manager, or pm2. However, developers can increase the default number of threads in the libuv thread pool. The server operating system (OS) is likely to distribute these threads across multiple cores. Another problem is that long-lasting computations and other CPU-bound tasks freeze the entire event-loop until completion.[citation needed]

Node.js uses libuv to handle asynchronous events. Libuv is an abstraction layer for network and file system functionality on both Windows and POSIX-based systems such as Linux, macOS, OSS on NonStop, and Unix.

The core functionality of Node.js resides in a JavaScript library. The Node.js bindings, written in C++, connect these technologies to each other and to the operating system.

5.6.2.2 V8

V8 is the JavaScript execution engine which was initially built for Google Chrome. It was then open-sourced by Google in 2008. Written in C++, V8 compiles JavaScript source code to native machine code at runtime. As of 2016, it also includes Ignition, a bytecode interpreter.

5.6.2.3 PACKAGE MANAGEMENT

npm is the pre-installed package manager for the Node.js server platform. It installs Node.js programs from the npm registry, organizing the installation and management of third-party Node.js programs. Packages in the npm registry can range from simple helper libraries such as Lodash to task runners such as Grunt.

5.6.2.4 UNIFIED API

Node.js can be combined with a browser, a database that supports JSON data (such as Postgres, MongoDB, or CouchDB) and JSON for a unified JavaScript development stack. With the adaptation of what were essentially server-side development patterns such as MVC, MVP, MVVM,

etc., Node.js allows the reuse of the same model and service interface between client side and server side.

5.6.2.5 EVENT LOOP

Node.js registers with the operating system so the OS notifies it of connections and issues a callback. Within the Node.js runtime, each connection is a small heap allocation. Traditionally, relatively heavyweight OS processes or threads handled each connection. Node.js uses an event loop for scalability, instead of processes or threads. In contrast to other event-driven servers, Node.js's event loop does not need to be called explicitly. Instead, callbacks are defined, and the server automatically enters the event loop at the end of the callback definition. Node.js exits the event loop when there are no further callbacks to be performed.

5.6.2.6 WEB ASSEMBLY

Node.js supports WebAssembly and as of version 14.x has an experimental support of WASI, the WebAssembly System Interface. [10]

5.7 SOLIDITY COMPILER

5.7.1 VERSION

Solidity versions follow semantic versioning and in addition to releases, nightly development builds are also made available. The nightly builds are not guaranteed to be working and despite best efforts they might contain undocumented and/or broken changes. We recommend using the latest release. Package installers below will use the latest release.

5.7.2 REMIX

Access Remix online, you don't need to install anything. If you want to use it without connection to the Internet, go to https://github.com/ethereum/remix-live/tree/gh-pages and download the .zip file as explained on that page.

5.7.3 NPM

Use npm for a convenient and portable way to install solcjs, a Solidity compiler. The solcjs program has fewer features than the ways to access the compiler described further down this page. The Using the Commandline Compiler documentation assumes you are using the full-featured compiler, solc. The usage of solcjs is documented inside its own repository.

Note: The solc-js project is derived from the C++ solc by using Emscripten which means that both use the same compiler source code. solc-js can be used in JavaScript projects directly (such as Remix). Please refer to the solc-js repository for instructions.

npm install -g solc

5.7.4 DOCKER

We provide up to date docker builds for the compiler. The stable repository contains released versions while the nightly repository contains potentially unstable changes in the develop branch.

docker run ethereum/solc:stable –version

Currently, the docker image only contains the compiler executable, so you have to do some additional work to link in the source and output directories.

5.7.5 BINARY PACKAGES

We also have PPAs for Ubuntu, you can get the latest stable version using the following commands:

sudo add-apt-repository ppa:ethereum/ethereum

sudo apt-get update

sudo apt-get install solc

The nightly version can be installed using these commands:

sudo add-apt-repository ppa:ethereum/ethereum

sudo add-apt-repository ppa:ethereum/ethereum-dev

sudo apt-get update

sudo apt-get install solc

We are also releasing a snap package, which is installable in all the supported Linux distros. To install the latest stable version of solc:

sudo snap install solc

If you want to help testing the latest development version of Solidity with the most recent changes, please use the following:

sudo snap install solc –edge

Arch Linux also has packages, albeit limited to the latest development version:

pacman -S solidity

We distribute the Solidity compiler through Homebrew as a build-from-source version. Prebuilt bottles are currently not supported.

brew update

brew upgrade

brew tap ethereum/ethereum

brew install solidity

If you need a specific version of Solidity you can install a Homebrew formula directly from Github.

View solidity.rb commits on Github.

Follow the history links until you have a raw file link of a specific commit of solidity.rb.

Install it using brew:

brew unlink solidity

Install 0.4.8

brew install

Gentoo Linux also provides a solidity package that can be installed using emerge:

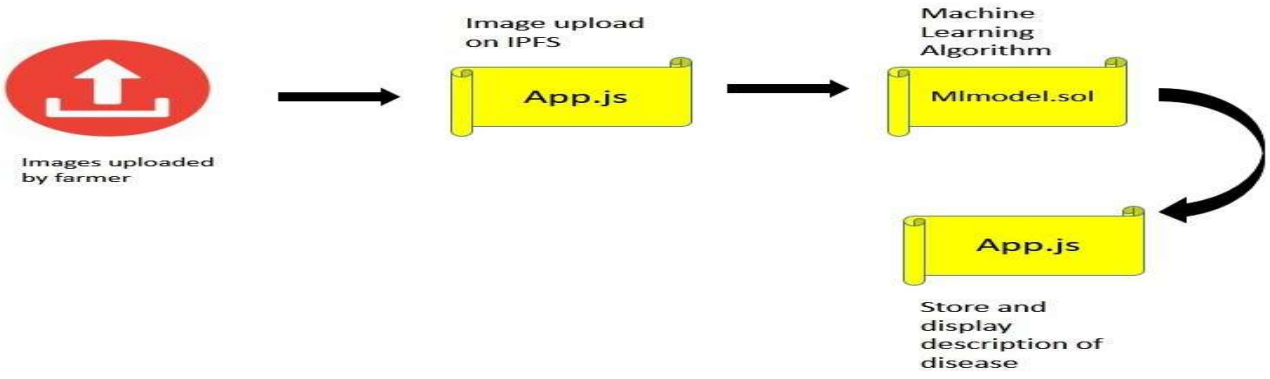
emerge dev-lang/solidity

6. SYSTEM DESIGN

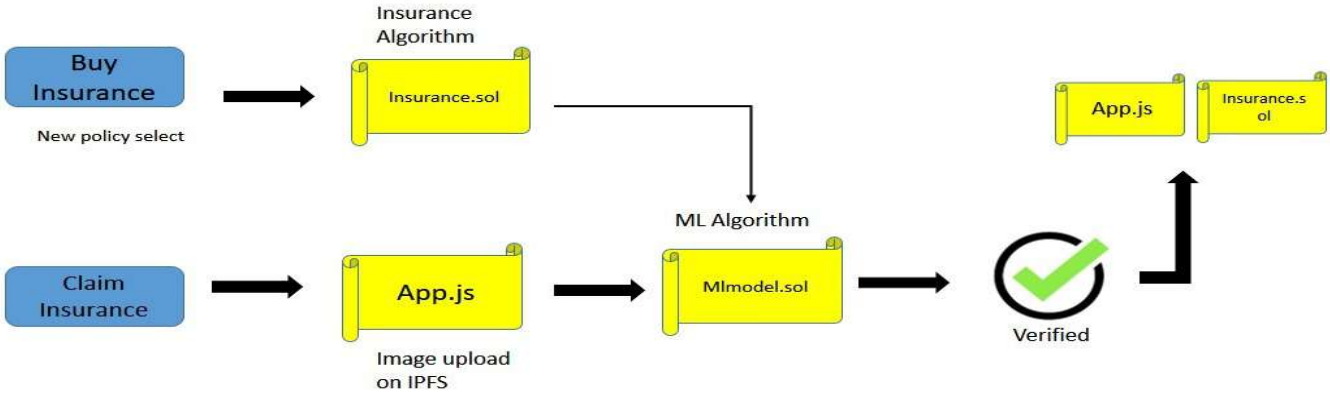
6.1 MODEL FLOW DIAGRAM

The basic flow of our decentralized model in which we have a two smart contract first one is mlmodel.sol and second one is insurance.sol .mlmodel.sol interact with our machine learning model and every image we pass that can predict the disease in the leaf and the information generated will be stored in a variable named predicted and insurance.sol deals with all the forms of insurance and through this smart contract we take a new insurance for any crop and we can also claim insurance of any existing crop.

We have also a app.js and this is java script file and its deals with ipfs because for this we need a image and storage capacity of blockchain is not sufficient for store an image and it is also very costly or expensive so we have used one of the alternative is ipfs and in ipfs when we upload an image to ipfs it automatically get stored in ipfs server and generated hash and the generated hash can easily be passed to machine learning to predict the disease.



In the first flow diagram firstly we have to upload the images .The uploaded images will further go to app.js which is java script file and after that it will get uploaded to ipfs and hash will be generated. The generated hash will go to mlmodel.sol and the generated information will go back to java script file and then disease information will be display.



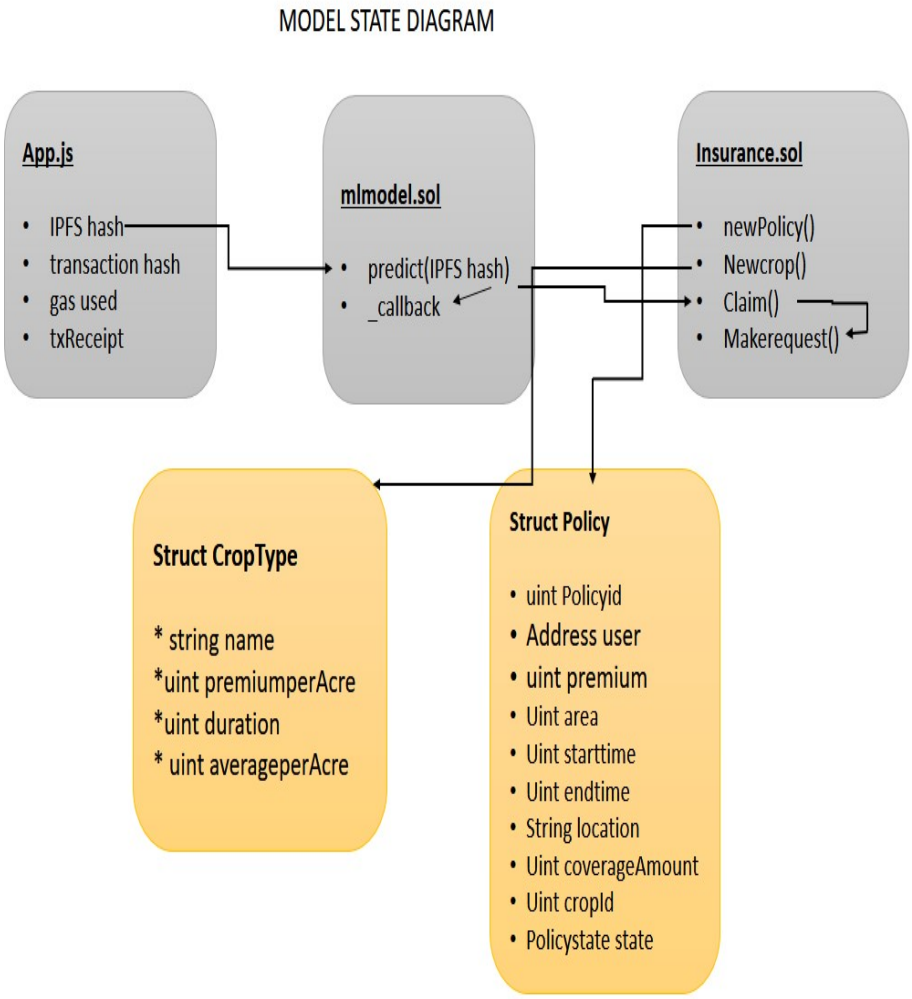
In the second flow diagram we have two option first one is buy insurance and second one is claim insurance. In buy insurance we select a new policy first through insurance algorithm we purchase a new insurance and after that we will go to machine learning algorithm to see whether a disease in being predicted or not and after it we will also see weather there is already a disease or not and if there is no disease then we will not give insurance. In claim insurance under this we upload images through ipfs to app.js after this machine learning algorithm will verify whether there is a disease present or not . After verification app.js will display information regarding verification and insurance.sol will display information regarding weather insurance is claimed or not.

6.2 MODEL STATE DIAGRAM

The state flow diagram of the model has three primary files that influence the workflow of the model. It includes two smart contracts, namely "mlmodel.sol" and "insurance.sol" and a javascript file saved as "app.js". When someone uploads the images, through the "app.js" file it'll be stored on the IPFS and a cryptographic hash is generated. The hash is then passed to the "mlmodel.sol" file, there the hash will be used as a parameter for the predict function and then it'll return a result with the callback function. Then the result will proceed to the claim function inside "insurance.sol" file. When the condition of the claim function is fulfilled it'll call the makerequest function, which will determine the claim amount.

Inside the "insurance.sol" file there's a newPolicy function that'll ask the user's details stored in the Policy structure. The details include Policy id, Adress, Premium Amount, Coverage Area, Start and End time, Crop id, Policy state, etc. The end time is calculated based on the crop type

(Rabi and Kharif). Moreover, the user can add a new crop with the newcrop function built in the "insurance.sol" It has structure named as cropType which will store the Crop name, Premium per acre, Duration, etc.



7. PROPOSED MODEL

7.1 DECENTRALISED CROP INSURANCE

In the modern age, the Blockchain revolution has popularised the decentralization of every application. Farmers frequently face the problems of crop damage because of several plant diseases. Many insurance companies are covers these factors, but the farmers that are insured face challenges to get their coverage outlay from the insurance companies.

Furthermore, the insurance corporations always act as a middleman in all this process. To remove the middleman and immediately and settle insurance claims by the user this decentralized insurance system is made. The Dapp has various options to buy the insurance, Users can opt for their policy by specifying the type of crop, start time, area and all other necessary details.

The end time is automatically calculated depending on the type of crop (i.e. Rabi or Kharif). The users pay the required premium fee to the smart contract using the p2p network system. After buying, the users will get a unique Policy ID which needs to be preserved. In case of crop damage, the users can use the claim option and enter their Policy ID. In order to calculate the coverage amount, some parameters have been defined in the contract depending on the type of disease.

7.2 MACHINE LEARNING MODEL

In order to get the prediction of the crop diseases from their images, there was a need to have a machine learning model at the backend. Now, this model was built up in python language with the help of a dataset obtained from the kaggle website.

The objectives of the dataset were to train a model using images of training dataset to 1) Accurately classify a given image from testing dataset into different diseased category or a healthy leaf; 2) Accurately distinguish between many diseases, sometimes more than one on a single leaf; 3) Deal with rare classes and novel symptoms; 4) Address depth perception—angle, light, shade, physiological age of the leaf; and 5) Incorporate expert knowledge in identification, annotation, quantification, and guiding computer vision to search for relevant features during learning.

This dataset contained approximately 3642 leaf images and their labels were under four classes that is leaves that are healthy, leaves with multiple diseases, leaves with rust and leaves with scabs.

After the loading of this dataset, the images had to be preprocessed to obtain all the important features of the leaves. Scaling, normalisation of pixels, edge detection and splitting and merging of respective colour channels was done. After which, the processed images over here were appended into a different array.

Now this array was splitted into training, validation and testing datasets. Thereafter, a sequential model, consisting of densenet121, global average pooling2d and a dense layer of 4 neurons, was defined. Here, in the densenet121 layer approximately 29089792 pre trained imagenet weights were used in order to get a high accuracy.

The model summary was as follows:

Layer (type)	Output Shape	Param #
densenet121 (Functional)	(None, 6, 6, 1024)	7037504
global_average_pooling2d (Gl	(None, 1024)	0
dense (Dense)	(None, 4)	4100

Total params: 7,041,604
Trainable params: 6,957,956
Non-trainable params: 83,648

The model obtained was compiled using rmsprop optimizer and categorical crossentropy loss. Now, the training dataset was then made to pass via the sequential model for about 200 epochs, obtaining a training categorical accuracy of 1 at the end. Then, the prediction was done and after that a classification report was obtained, wherein, the model gave the f1-score, by the weighted average of all the four classes, of 0.98. Thereby, proving to give an accuracy of about 98 percent in the predictions of the foliar disease.

At later stages, this model was turned into an API and then oraclized into the smart contract to connect it further to the proposed decentralised application.

7.3 TURNING MACHINE LEARNING MODEL TO AN API

Web APIs have made it easy for cross-language applications to work well. If a frontend developer needs to use your ML Model to create an ML-powered web application, they would just need to get the URL Endpoint from where the API is being served. In simple words, an API is a (hypothetical) contract between 2 software saying if the user software provides input in a pre-defined format, the later with extending its functionality and provide the outcome to the user software.

Essentially, APIs are very much like web applications, but instead of giving the styled HTML page, APIs tend to return data in a standard data-exchange format such as JSON, XML, etc.

Once the developer has the desired output they can style it whatever the way they want. There are many popular ML APIs available out there as well. IBM Watson's ML API which is capable of the following:

• Machine Translation - Helps translate text in different language pairs.

• Message Resonance – To find out the popularity of a phrase or word with a predetermined audience.

• Question and Answers - This service provides direct answers to the queries that are triggered by primary document sources.

• User Modelling – To make predictions about the social characteristics of someone from a given text.

7.3.1 FLASK

Web service is a form of API only that assumes that an API is hosted over a server and can be consumed. Flask is a web service development framework in Python. It is not the only one in Python, there a couple of others as well such as Django, Falcon, Hug, etc. Flask is very minimal.

Flask framework comes with an inbuilt light-weighted web server that needs minimal configuration, and it can be controlled from the Python code. Flask will load the already persisted model into memory when the application starts. Moreover, it'll create an API endpoint that takes input variables, transform them into the appropriate format, and returns predictions.

Flask is used with our Densenet machine learning model to create an API for further use in smart contracts. It'll be used as a core state to utilize our model.

7.4 ORACLIZING THE API

The smart contracts built are like a closed box, the code inside it cannot communicate with the external world on their own. However, it is really important that the smart contracts communicate with the machine learning API, in order to predict the diseases via the user interface for the images uploaded by the farmers. One option for this communication is by using services such as Oraclize. It acts as a data carrier and a reliable connection between APIs and the decentralised application.

To use Oraclize on testrpc and truffle, ethereum-bridge needs to be installed and this is a very essential requirement for the service. Thereafter, the oracliseAPI is imported from the github

link in the smart contract wherein further, a contract is initialized and an event is generated with a description as parameter.

Then a function is declared for the prediction which is of payable type and here the access specifier is defined public. Now here, the generated event and a predefined function oraclize query is called with a parameter "URL" as the interaction made here is by the URL of the ML model API. The URL of the API is specified as the second parameter of the function under the json command, owing to the fact that the API is written in the json format at the backend mapped with the keys and values.

When the function predict communicates with the API, it gets a unique queryId, the result of the prediction and a proof of the authentication. After this, a callback function is called by itself, which contains the parameters like queryId, prediction result and the proof. This function also checks whether the call back address is similar to that of the message sender or not, once it gets authenticated, the predicted result is sent to the insurance smart contract.

The insurance smart contract further checks the different classes of the predicted result and accordingly decides the coverage amount that the farmer would get. Simultaneously, the information of the predicted disease also gets displayed on the user interface to let them know about the disease that their crop is subjected to or may suffer with.

7.5 IPFS INTERACTION WITH SMART CONTRACT

Now, as the user uploads an image on the decentralised application, the image needs to get stored at some place within the application, and as this is a decentralised application there is no server for storage and the data gets stored on the blockchain ledger. However, this is a time consuming task, therefore, there was a need for the interplanetary file system(IPFS) where the data is distributed over the peer-to-peer network and does not take much time for uploading or retrieval of data.

When dealing with IPFS storage, one can directly install IPFS on their systems and thereafter can initialise the daemon and could use the website to upload their file and from there only, it could be accessed as well.

However, to make the application a single place for all the services, it needs to be integrated with the user interface and for this one needs to unbox react firstly at the directory of workspace and thereafter the ipfs-api needs to be installed and after that the ipfs-infura.io is introduced as the host for the data uploading purposes. Now, another javascript file is created to import the web3 and to fetch the abi of the solidity files to interact with the front end.

Hence, whenever the user uploads an image on the application, a capture file event is generated and here the image file is read as the array buffer. The console of the user interface obtains the web3 account and metamask transaction account information. The file is then uploaded on the IPFS, which further generates a unique hash value, namely the IPFS hash, for the image file. This hash is sent to the user interface which gets displayed on the screen for the user's convenience to further check on the IPFS website itself for their image. In addition to the IPFS hash, the ethereum contract address and the transaction hash also get displayed for future correspondence and reference.

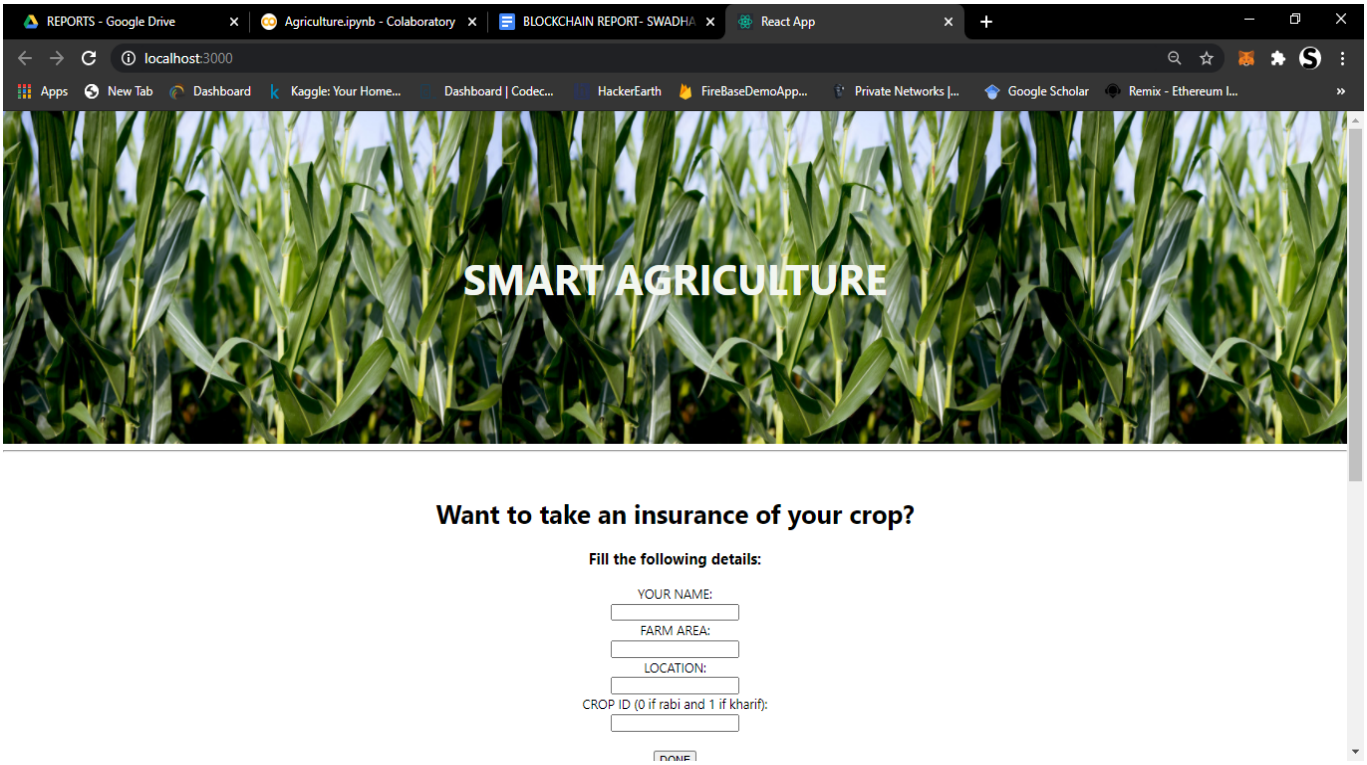
8. RESULTS ANALYSIS

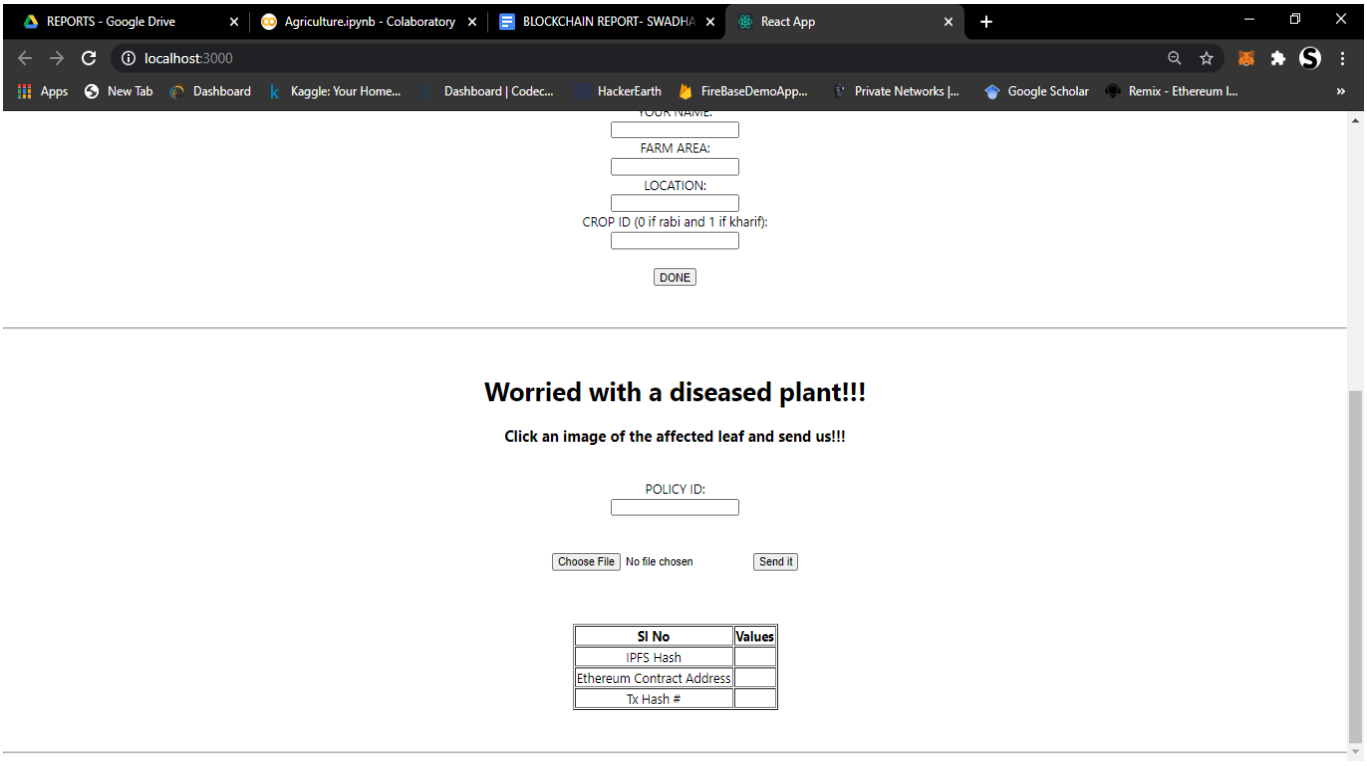
8.1 MACHINE LEARNING MODEL RESULTS

	precision	recall	f1 score	support
0	0.93	0.93	0.95	280
1	0.94	0.96	0.93	234
2	0.95	0.97	0.99	318
3	0.92	0.92	0.98	260
micro avg	0.97	0.96	0.96	1092
macro avg	0.92	0.93	0.94	1092
weighted avg	0.97	0.99	0.98	1092
samples avg	0.99	0.96	0.95	1092

A convolutional neural network is a deep learning algorithm which can take in an input image, assign importance to various objects in the image and be able to differentiate one from other. It is one of the main categories to do images recognition, images classifications. CNN image classifications takes an input image, process it and classify it under certain categories. In this model firstly convolutional neural network is used but this convolutional neural network doesn't provide good accuracy apart from this in place of convolutional network DenseNet network is used and this DenseNet network provide very good accuracy because DenseNet network concatenates (.) the output of the previous layer with the future layer. The DenseNet is divided into DenseBlocks where a number of filters are different, but dimensions within the block are the same. It is one of the latest neural networks for visual object recognition. It's quite similar to RenseNet but has some fundamental differences.

8.2 FRONTEND


REPORTS - Google Drive
Agriculture.ipynb - Colaboratory
BLOCKCHAIN REPORT- SWADHA
React App
localhost:3000
Apps
New Tab
Dashboard
Kaggle: Your Home...
Dashboard | Codec...
HackerEarth
FireBaseDemoApp...
Private Networks |...
Google Scholar
Remix - Ethereum I...
SMART AGRICULTURE
Want to take an insurance of your crop?
Fill the following details:
YOUR NAME:
FARM AREA:
LOCATION:
CROP ID (0 if rabi and 1 if kharif):
DONE


REPORTS - Google Drive
Agriculture.ipynb - Colaboratory
BLOCKCHAIN REPORT- SWADHA
React App
localhost:3000
Apps
New Tab
Dashboard
Kaggle: Your Home...
Dashboard | Codec...
HackerEarth
FireBaseDemoApp...
Private Networks |...
Google Scholar
Remix - Ethereum I...
FARM AREA:
LOCATION:
CROP ID (0 if rabi and 1 if kharif):
DONE
Worried with a diseased plant!!!
Click an image of the affected leaf and send us!!!
POLICY ID:
Choose File
No file chosen
Send it
SI No
Values
IPFS Hash
Ethereum Contract Address
Tx Hash #

The frontend of the proposed decentralised application provides various functionalities to the farmers. One could insure his crop through the application for a certain time period and if within that time period his crop gets harmed, he could easily upload the images of the affected leaves and if the machine learning model finds a damage to the crops, his damage would be covered by giving a specific coverage amount based on the disease caused.

The frontend here is developed using react scripts, which had to be installed on the system via the command- npm install react-scripts. The versions used were as follows: React: 16.11.0 React-dom: 16.11.0 React-scripts: 3.2.0

The frontend was linked with the ipfs-api in order to directly upload the images via ipfs and obtain the ipfs hash, ethereum contract address and the transaction hash and then the prediction is made whose result gets displayed on the screen.

The frontend also interacts with the insurance smart contract written in solidity to ensure the insurance functionalities effectively.

9. DISCUSSION

9.1 FUTURE WORK

9.1.1 ENHANCING THE MODELS WITH MORE DATASETS

Currently the dataset used is very limited in dynamics. There are numerous of crops and plants of dataset. Future plans is include adding specimens of leaf's of various plants and crops. Future dataset be cereals, pulses, rice and various fruits. These specimens will form future dataset and after that dataset will be trained in our model.

Data set includes various types of leaves. Some leaves show healthy characteristics and some shows some kind of disease. With help of model leaves can be examined, weather they are healthy or not. And if there is some kind of disease present it can also predict whether it is fungal or parasitic. For making model more accurate even sub-categories specimens will be added to dataset, like there are various varities of rices and various varities of oranges plant. Addition of various sub-category leaves in a particular crop/plant. Thus, more samples will lead to more accurate model in future.

9.1.2 ADDING ANOTHER LEDGER FOR SPECIALISTS DETERMINING THE TREATMENT FOR THE DISEASES

Currently the examination of diseases is going basically through machine learning model. Image of apple leaves are examined by model for predicting diseases. This is current scenario of examination. Future plans includes hiring a specialist who will be a researcher for tackling the more complex unpredictable diseases. Model has a limitation because it can only work on basis of dataset.

If a rare disease in encountered who's sample is not yet available , model will fail there. Here, a specialist will examine sample physically . If it is a new disease then it will be his/her task to determine its dimension and type of diseases and specialist will also charge the fee. Through this we will come to know about the disease. The specialist will be paid for his work.

9.1.3 ADDING SERVICES FOR CASES SUCH AS FLOODS AND DROUGHTS

Currently insurance facility is available only for the apple plant in case of disease present. But future plans will broaden insurance dimensions. Like in the future there will be a insurance scheme for the stored foods too. If stored food gets damaged in period of storage, in that case crop damages due to natural disasters like flood and droughts. Broad insurance policy in future will greatly help farmers in future.

9.2 IMPLICATION

The proposed model enables the decentralised application to store the data on peer-to-peer networks implementing interplanetary file systems and a hypermedia protocol to make the web faster, safer and much more transparent. It abolishes the need of third party intermediates and a central server.

The traditional conventions have always laid focus on central authority to have the supreme decisive powers. There are cases where such authorities get biased and the user is denied his rights. Moreover, these insurance processes become so hassle that most of the farmers do not feel free to get insured due to lack of services.

However, the proposed model would provide a user friendly service which would help the farmers to take or claim an insurance just through some clicks. It would enforce transparency throughout the processes and there will be no cases of biasing as the ML Model will itself act as the proof of claiming insurance. The ML model working here is highly accurate and precise and could easily identify even the diseases that could not be visualised by a naked eye.

10. CONCLUSION

In this model, a decentralized assisted plant disease prediction is proposed using deep convolutional neural networks. It works with a blending of machine learning and Blockchain technology. Apparently, the model has been able to predict 4 major diseases in apple plants with good accuracy, as well as gives the prompt to opt for decentralized insurance. The model enables any individual to help protect their crops. It has a simple web user interface, so It can be used by farmers all around the earth. In the future, the developed system will be linked to the National CropPest Management System to provide information on infection risks such as risk-based on the crop, mycelial growth rate, disease development speed, germination rate, and disease outbreak quantity.

The model intends to spread awareness among the masses by lessening the burden on the shoulders of the farmers and making the crop insured of losses. This would help the farmers as a boon in bad times.

The model is highly precise in detecting the diseases and this would help in reducing the wastage of crops by pests, pathogens and diseases. At the same time, being distributed and decentralised, it also maintains the security and integrity of data. Moreover, the model would also provide hassle free solutions to the farmers as compared to the traditional insurance methods.

References

[1] Imran Bashir. *Mastering Blockchain: Distributed ledger technology, decentralization, and smart contracts explained*. Packt Publishing Ltd, 2018.

[2] Daniel P Bebber, Timothy Holmes, and Sarah J Gurr. The global spread of crop pests and pathogens. *Global Ecology and Biogeography*, 23(12):1398–1407, 2014.

[3] Gerald A Carlson. A decision theoretic approach to crop disease prediction and control. *American Journal of Agricultural Economics*, 52(2):216–223, 1970.

[4] Cory Gackenheimer. *Introduction to React*. Apress, 2015.

[5] Adem Efe Gencer, Soumya Basu, Ittay Eyal, Robbert Van Renesse, and Emin Gün Sirer. Decentralization in bitcoin and ethereum networks. In *International Conference on Financial Cryptography and Data Security*, pages 439–457. Springer, 2018.

[6] Garrick Hileman and Michel Rauchs. Global cryptocurrency benchmarking study. *Cambridge Centre for Alternative Finance*, 33:33–113, 2017.

[7] Muqaddas Naz, Fahad A Al-zahrani, Rabiya Khalid, Nadeem Javaid, Ali Mustafa Qamar, Muhammad Khalil Afzal, and Muhammad Shafiq. A secure data sharing platform using blockchain and interplanetary file system. *Sustainability*, 11(24):7054, 2019.

[8] Michael Nofer, Peter Gomber, Oliver Hinz, and Dirk Schiereck. Blockchain. *Business & Information Systems Engineering*, 59(3):183–187, 2017.

[9] Aqeel-ur Rehman. *Smart Agriculture: An Approach towards Better Agriculture Management*. 02 2015.

[10] Mithun Satheesh, Bruno Joseph D'mello, and Jason Krol. *Web development with MongoDB and NodeJs*. Packt Publishing Ltd, 2015.

[11] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.