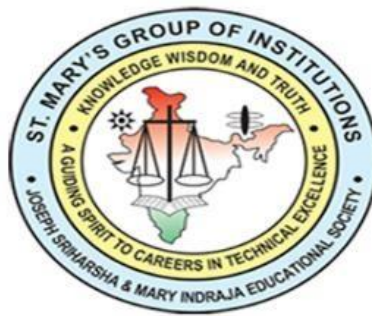


A PROJECT REPORT
ON
Blockchain E-Voting Done Right: Privacy and
Transparency with Public Blockchain
Submitted in partial fulfillment for the Award of Credits To
IV-II FOR PROJECT WORK
IN BACHELOR OF TECHNOLOGY
IN
ARTIFICIAL INTELLIGENCE AND DATA SCIENCE



Submitted By:

ATIPAMULA CHAKRAVARTHI	[20BJ1A5408]
SHAIK ILIAZ	[20BJ1A5456]
JAYAVARAPU SRIKANTH	[20BJ1A5430]
BATULA ANIL KUMAR	[20BJ1A5411]

Under the esteemed guidance of

S.PRASAD, M.Tech
ASSISTANT PROFESSOR
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ST.MARY'S GROUP OF INSTITUTIONS GUNTUR
(Affiliated to JNTU KAKINADA, Approved by AICTE, NEWDELHI & Accredited by 'NAAC')
CHEBROLU (V&M), GUNTUR DIST., ANDHRA PRADESH, INDIA, PIN: 522212,

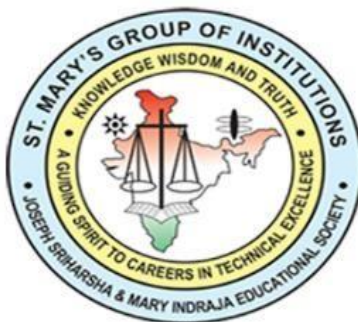
2020-2024

ST.MARY'S GROUP OF INSTITUTIONS GUNTUR

(Affiliated to JNTU Kakinada, Approved by AICTE, NEW DELHI & Accredited by 'NAAC')

CHEBROLU (V&M), GUNTUR DIST., ANDHRA PRADESH, INDIA, PIN: 522212

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the project report entitled “**Blockchain E-voting Done Right: Privacy and Transparency with Public Block chain**” is the bonafied project work carried out by **ATIPAMULA CHAKRAVARTHI [20BJ1A5408]**, **SHAIK ILIYAZ [20BJ1A5456]**, **SJAYAVARAPU SRIKANTH [20BJ1A5430]**, **BATHULA ANIL KUMAR [20BJ1A5411]** during the academic year **2020-2024**, in partial fulfillment of the requirements to the award of the Credits in IV-II of **Bachelor of Technology In Artificial Intelligence and Data Science** from **St. Mary's Group Of Institutions Guntur** of **Jawaharlal Nehru Technological University, Kakinada**.

PROJECT GUIDE

S.PRASAD M.Tech
Assistant professor

HEAD OF THE DEPARTMENT

Dr. S S N ANJANEYULU M.Tech, Ph.D
Associate professor

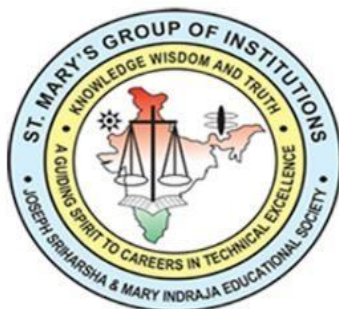
EXTERNAL EXAMINER

ST.MARY'S GROUP OF INSTITUTIONS GUNTUR

(Affiliated to JNTU, KAKINADA, Approved by AICTE, NEWDELHI & Accredited by 'NAAC')

CHEBROLU (V&M), GUNTUR DIST., ANDHRA PRADESH, INDIA, PIN: 522212

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



DECLARATION

We, **ATIPAMULA CHAKRAVARTHI [20BJ1A5408]**, **SHAIK ILIYAZ [20BJ1A5456]**, **JAYAVARAPU SRIKANTH [20BJ1A5430]**, **SBATHULA ANIL KUMAR [20BJ1A5411]** Students of B. Tech, ST.MARY'S GROUP OF INSTITUTIONS GUNTUR, Jawaharlal Nehru Technological University Kakinada, A.P, do hereby Declare that the Project Report Entitled **“Blockchain E-voting Done Right : Privacy and Transparency with Public Block chain”** is the Genuine Work carried out at ST.MARY'S GROUP OF INSTITUTIONS GUNTUR, under the Guidance of **S.PRASAD, M.TECH**, Department Of Computer Science and Engineering, ST.MARY'S GROUP OF INSTITUTIONS GUNTUR. We declare that the Work embodied in the thesis has not been submitted for the award of credits for Degree or Diploma of this or any other University.

ATIPAMULA CHAKRAVARTHI [20BJ1A5408]

SHAIK ILIYAZ [20BJ1A5456]

JAYAVARAPU SRIKANTH [20BJ1A5430]

BATHULA ANIL KUMAR [20BJ1A5411]

ACKNOWLEDGEMENT

On the occasion of presenting the dissertation, We would like to thank the **Almighty** for providing strength and power by his enormous blessings to overcome all the hurdles and hindrances during this project work. First & Foremost, We would like to express my sincere gratitude to our guide **S PRASAD, M.TECH**, Department of Computer Science And Engineering, St. Mary's Group of Institutions Guntur, for the continuous support of our B. Tech study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for our B. Tech study. We also thank Chairman Sir **Rev. Dr. K.V.K. RAO garu**, St. Mary's Group of Institutions Guntur for providing necessary facilities to carry out this research work. Besides our advisor, we would like to thank **Dr. B. PENCHALAIAH, Principal, Dr. S. APPARAO, Director & Dr. S S N ANJANEYULU, Head of The Department**, St. Mary's Group of Institutions Guntur, for their guidance, support, encouragement and valuable suggestions. We also express my deepest sense of gratitude to all Lab technicians of SMGG. We would like to thank all the teaching and non-teaching staff of who were very helpful in completion of our project work successfully. We would extend my heartfelt thanks to our family: our parents for giving birth to me at the first place and supporting me spiritually throughout my life. We are fortunate to have friends and classmates, who constantly helped me throughout the period, this study was carried out. I extend my hearty gratitude to all those who have directly or indirectly helped me during this research. Thanks to one and all....

INDEX

Table Of Contents	Page No
Abstract	
Chapter 1: Introduction	1
1.1 Objective of the project	2
Chapter 2: Literature Survey	4
Chapter 3: System Analysis	
3.1 Existing System	7
3.2 Proposed System	8
3.3 Modules	9
3.4 Process Model Used With Justification	10
3.5 Software Requirements	19
3.6 External Interface Requirements	21
Chapter 4: System Design	22
4.1 Class Diagram	23
4.2 Use case Diagram	24
4.3 Sequence Diagram	25
4.4 Component Diagram	28
4.5 Deployment Diagram	29
4.6 Activity Diagram	31
4.7 Data Flow Diagram	33
Chapter 5: Implementation	35
5.1 Python	
5.2 Sample Code	39

Chapter 6: Testing	43
Chapter 7: Screenshots	46
Chapter 8: Conclusion	56
Chapter 9: References	57

ABSTRACT

ABSTRACT

Since then, there have been several forms of voting. Most countries still use paper ballots to cast their votes. Electronic voting methods have just become widespread in the previous 10 years and still have several unsolved issues. Security, legitimacy, transparency, openness, dependability, and usefulness are the most pressing concerns with electronic voting. This is a field in which Estonia is a pioneer and a leader. However, when it comes to blockchain, the options are limited. It is possible to address all of the issues listed with the help of blockchain technology, which also provides advantages like as immutability and decentralisation. For e-voting, the main issues with blockchain technology are the lack of testing and comparison, as well as the concentrate on a single topic. A blockchain-based electronic voting system that can be used for any type of vote is described in this paper. All procedures may be completed on the blockchain. Decentralization ensures that once voting begins, the platform is completely independent and cannot influence the outcome of the vote. Homomorphic encryption ensures that the voters' identities are protected. Our solution was tested and compared on three different blockchains. The findings show that both public and private blockchains are available at a relatively low speed. Because of the homomorphic encryption used to protect voter data, we've developed a new approach that completely decentralises the management of the e-voting platform using blockchains.

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION

The topic of e-voting systems is still at an early stage of development. We have chosen this domain not only for its recency but also because there are not many solutions that address problems of e-voting. Nowadays, popularity grows also in the development of e-Government. However, such a system is not feasible if basic services for citizens such as elections do not become electronic. "E-voting is one of the key public sectors that can be transformed by blockchain technology". Hand by hand with e-voting come also new challenges, which need to be addressed. One of them is e.g. securing the elections, which needs to be at least as safe as the classic voting systems with ballots. That is why we have decided to create safe elections in which voters do not have to worry about someone abusing the electoral system. In recent years blockchain is often mentioned as an example of secure technology used in an online environment. Our e-voting system uses blockchain to manage all election processes. Its main advantage is that there is no need for confidence in the centralized authority that created the elections. This authority cannot affect the election results in our system. Another challenge in e-voting is the lack of transparency in the functioning of the system, leading to a lack of confidence in voters. This problem is solved by blockchain in a way of total transparency that allows everyone to see the stored data and processes such as how these data are handled. In the field of security, this technology is more suitable in every way than the classic e-voting platform without blockchain.

1.2 OBJECTIVE OF THE PROJECT

Some forms of voting have been here ever since. Mostly used form all over the world are paper ballots. Electronic voting schemes are being popular only in the last decade and they are still unsolved. E-voting schemes bring problems mainly regarding security, credibility, transparency, reliability, and functionality. Estonia is the pioneer in this field and may be considered the state of the art. But there are only a few solutions using blockchain. Blockchain can deliver an answer to all of the mentioned problems and furthermore bring some advantages such as immutability and decentralization. The main problems of technologies utilizing blockchain for e-voting are their focus on only one field or lack of testing and comparison. In this paper, we present a blockchainbased e-voting platform, which can be used for any kind of voting. It is fully utilized by blockchain and all processes can be handled within it. After the start of the voting, the platform behaves as fully independent and decentralized without possibilities to affect the voting process. The data are fully transparent, but the identity of voters is secured by homomorphic encryption. We have tested and compared our solution in three different blockchains. The results show, that both public and private blockchains can be used with only a little difference in the speed. The key novelty of our solution is a fully decentralized management of e-voting platformthrough blockchain, transparency of the whole process and at the same time security and privacy of the voters thanks to homomorphic encryption.

CHAPTER-2

LITERATURE SURVEY

“Block-chain-Enabled E-Voting,”

Blockchain-enabled e-voting (BEV) could reduce voter fraud and increase voter access. Eligible voters cast a ballot anonymously using a computer or Smartphone. BEV uses an encrypted key and tamper-proof personal IDs. This article highlights some BEV implementations and the approach's potential benefits and challenges.

“Voting Process with Block-chain Technology: Auditable Block-chain Voting System,”

There are various methods and approaches to electronic voting all around the world. Each is connected with different benefits and issues. One of the most important and prevalent problems is lack of auditing capabilities and system verification methods. Blockchain technology, which recently gained a lot of attention, can provide a solution to this issue. This paper presents Auditable Blockchain Voting System (ABVS), which describes e-voting processes and components of a supervised internet voting system that is audit and verification capable. ABVS achieves this through utilization of blockchain technology and voter-verified paper audit trail.

“Bitcoin: A Peer-to-Peer Electronic Cash System,”

A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer- to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof- of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longestchain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a

best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

“A Smart Contract for Boardroom Voting with Maximum Voter Privacy,”

We present the first implementation of a decentralized and self-tallying internet voting protocol with maximum voter privacy using the Blockchain. The Open Vote Network is suitable for boardroom elections and is written as a smart contract for Ethereum. Unlike previously proposed Blockchain e-voting protocols, this is the first implementation that does not rely on any trusted authority to compute the tally or to protect the voter’s privacy. Instead, the Open Vote Network is a selftallying protocol, and each voter is in control of the privacy of their own vote such that it can only be breached by a full collusion involving all other voters. The execution of the protocol is enforced using the consensus mechanism that also secures the Ethereum blockchain. We tested the implementation on Ethereum’s official test network to demonstrate its feasibility. Also, we provide a financial and computational breakdown of its execution cost.

“Efficient Fully Homomorphic Encryption from (Standard) LWE,”

We present a fully homomorphic encryption scheme that is based solely on the (standard) learning with errors (LWE) assumption. Applying known results on LWE, the security of our scheme is based on the worst-case hardness of "short vector problems" on arbitrary lattices. Our construction improves on previous works in two aspects: 1) we show that "somewhat homomorphic" encryption can be based on LWE, using a new re-linearization technique. In contrast, all previous schemes relied on complexity assumptions related to ideals in various rings. 2) We deviate from the "squashing paradigm" used in all previous works. We introduce a new dimension-modulus reduction technique, which shortens the ciphertexts and reduces the decryption complexity of our scheme, without introducing additional assumptions. Our scheme has very short ciphertexts and we therefore use it to construct an asymptotically efficient LWE- based single-server private information retrieval (PIR) protocol. The communication complexity of our protocol (in the public-key model) is $k \cdot \text{polylog}(k) + \log |\text{DB}|$ bits per single-bit query (here, A ; is a security parameter).

“Definitions and properties of zero knowledge proof systems,”

In this paper we investigate some properties of zero-knowledge proofs, a notion introduced by Goldwasser, Micali, and Rackoff. We introduce and classify two definitions of zero-knowledge: auxiliary-input zero-knowledge and blackbox-simulation zero-knowledge. We explain why auxiliary-input zero-knowledge is a definition more suitable for cryptographic applications than the original [GMR1] definition. In particular, we show that any protocol solely composed of subprotocols which are auxiliary-input zero-knowledge is itself auxiliary-input zero-knowledge. We show that blackbox-simulation zero-knowledge implies auxiliary-input zero-knowledge (which in turn implies the [GMR1] definition). We argue that all known zero-knowledge proofs are in fact blackbox-simulation zero-knowledge (i.e., we proved zero-knowledge using blackbox-simulation of the verifier). As a result, all known zero-knowledge proof systems are shown to be auxiliary-input zero-knowledge and can be used for cryptographic applications such as those in [GMW2]. We demonstrate the triviality of certain classes of zero-knowledge proof systems, in the sense that only languages in BPP have zero-knowledge proofs of these classes. In particular, we show that any language having a Las Vegas zero-knowledge proof system necessarily belongs to RP. We show that randomness of both the verifier and the prover, and nontriviality of the interaction are essential properties of (nontrivial) auxiliary-input zero-knowledge proofs.

“Ethereum: A secure decentralized generalized transaction ledger,”

The blockchain paradigm when coupled with cryptographically-secured transactions has demonstrated its utility through a number of projects, not the least being Bitcoin. Each such project can be seen as a simple application on a decentralized, but singleton, compute resource. We can call this paradigm a transactional singleton machine with shared-state. Ethereum implements this paradigm in a generalized manner. Furthermore it provides a plurality of such resources, each with a distinct state and operating code but able to interact through a message-passing framework with others. We discuss its design, implementation issues, the opportunities it provides and the future hurdles we envisage.

CHAPTER-3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM:

Some forms of voting have been here ever since. Mostly used form all over the world are paper ballots. Electronic voting schemes are being popular only in the last decade and they are still unsolved. E-voting schemes bring problems mainly regarding security, credibility, transparency, reliability, and functionality. Estonia is the pioneer in this field and may be considered the state of the art. But there are only a few solutions using blockchain. Blockchain can deliver an answer to all of the mentioned problems and furthermore bring some advantages such as immutability and decentralization. The main problems of technologies utilizing blockchain for e-voting are their focus on only one field or lack of testing and comparison.

Disadvantages of Existing System:

1. Less Security.

3.2PROPOSED SYSTEM:

In this paper, we present a blockchain based e-voting platform, which can be used for any kind of voting. It is fully utilized by blockchain and all processes can be handled within it. After the start of the voting, the platform behaves as fully independent and decentralized without possibilities to affect the voting process. The data are fully transparent, but the identity of voters is secured by homomorphic encryption. We have tested and compared our solution in three different blockchains. The results show, that both public and private blockchains can be used with only a little difference in the speed. The key novelty of our solution is a fully decentralized management of e-voting platform through blockchain, transparency of the whole process and at the same time security and privacy of the voters thanks to homomorphic encryption.

Advantages of Proposed System:

1. More Security.

3.3MODULES:

Modules Description:

Admin module: This user responsible to add new party and candidate details and can view party details and vote count. Admin login to system by using username as ‘admin’ and password as ‘admin’.

User Module: This user has to sign up with the application by using username as his ID and then upload his face photo which capture from webcam. After registering user can go for login which validate user id and after successful login user can go for cast vote module which execute following functionality

1.First user will be connected to his PC webcam and then image will be capture

Using OpenCV application will detect face and then using CNN application will

2.predict user identify and if user identity matched with CNN predicted face then

application will display all voting candidates list.

3.If user not casted vote then user can give vote to desire candidate by clicking link beside party name or candidate name.

4.Upon giving vote application will capture voter and candidate details and then encrypt the data and then store in Blockchain.

[illegible]

3.4 PROCESS MODEL USED WITH JUSTIFICATION

SDLC (Umbrella Model):

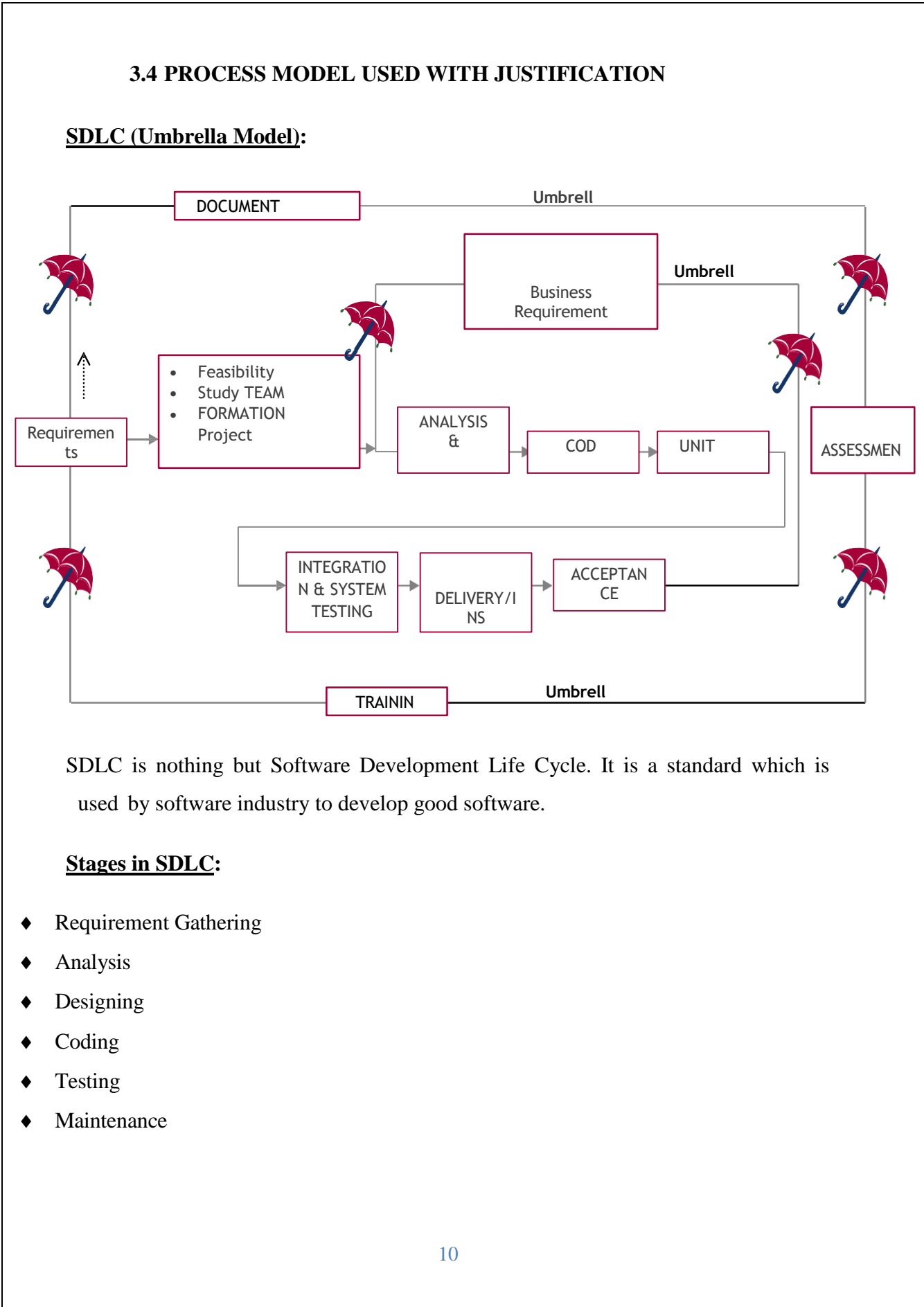
```
graph LR; Requirements[Requirements] --> Feasibility["Feasibility<br/>Study TEAM<br/>FORMATION Project"]; Feasibility --> Analysis[ANALYSIS &]; Analysis --> COD[COD]; COD --> UNIT[UNIT]; UNIT --> ASSESSMENT[ASSESSMEN]; ASSESSMENT --> ACCEPTANCE[ACCEPTAN CE]; ACCEPTANCE --> DELIVERY[DELIVERY / I NS]; DELIVERY --> TESTING[INTEGRATIO N & SYSTEM TESTING]; TESTING --> TRAINING[TRAININ]; TRAINING --> Requirements; Requirements -.->|Umbrell| DOCUMENT[DOCUMENT]; DOCUMENT -.->|Umbrell| BusinessRequirement[Business Requirement]; BusinessRequirement -.->|Umbrell| ASSESSMENT; ASSESSMENT -.->|Umbrell| TRAINING; TRAINING -.->|Umbrell| Requirements;
```

The diagram illustrates the SDLC (Umbrella Model) as a continuous cycle. It begins with 'Requirements' leading to a box containing 'Feasibility', 'Study TEAM', and 'FORMATION Project'. This leads to 'ANALYSIS &', followed by 'COD', 'UNIT', 'ASSESSMEN', 'ACCEPTAN CE', 'DELIVERY / I NS', and 'INTEGRATIO N & SYSTEM TESTING'. The cycle returns to 'Requirements' via 'TRAININ'. An 'Umbrell' layer connects 'DOCUMENT', 'Business Requirement', 'ASSESSMEN', 'TRAININ', and 'Requirements'.

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

Stages in SDLC:

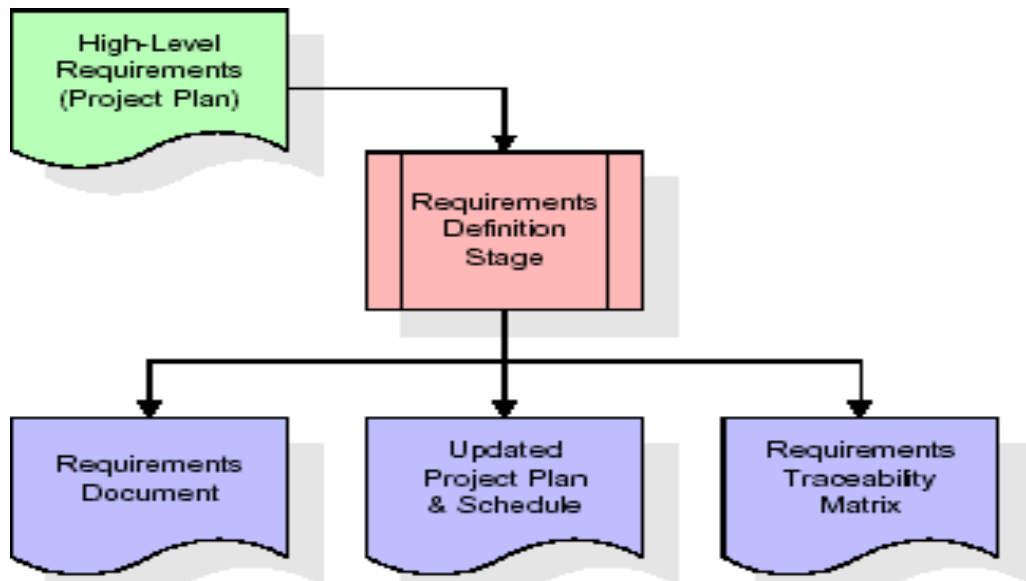
- ◆ Requirement Gathering
- ◆ Analysis
- ◆ Designing
- ◆ Coding
- ◆ Testing
- ◆ Maintenance

[illegible][illegible]

- [illegible]

Requirements Gathering stage:

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product

components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

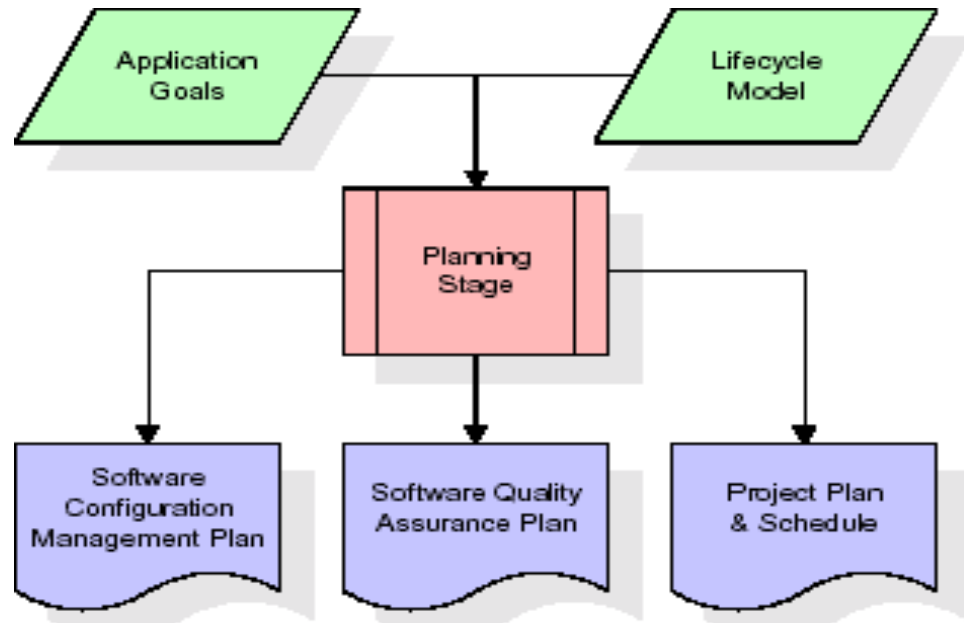
In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

- Feasibility study is all about identification of problems in a project.
- No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
- Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

Analysis Stage:

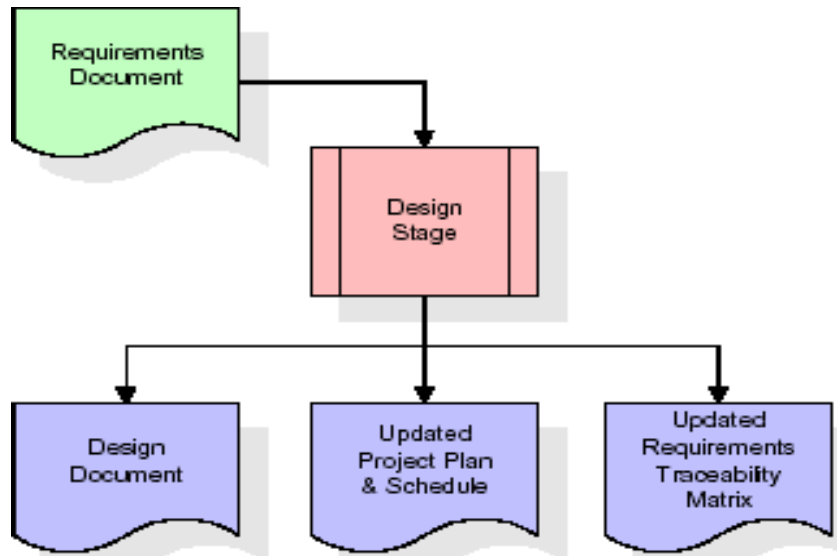
The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

Designing Stage:

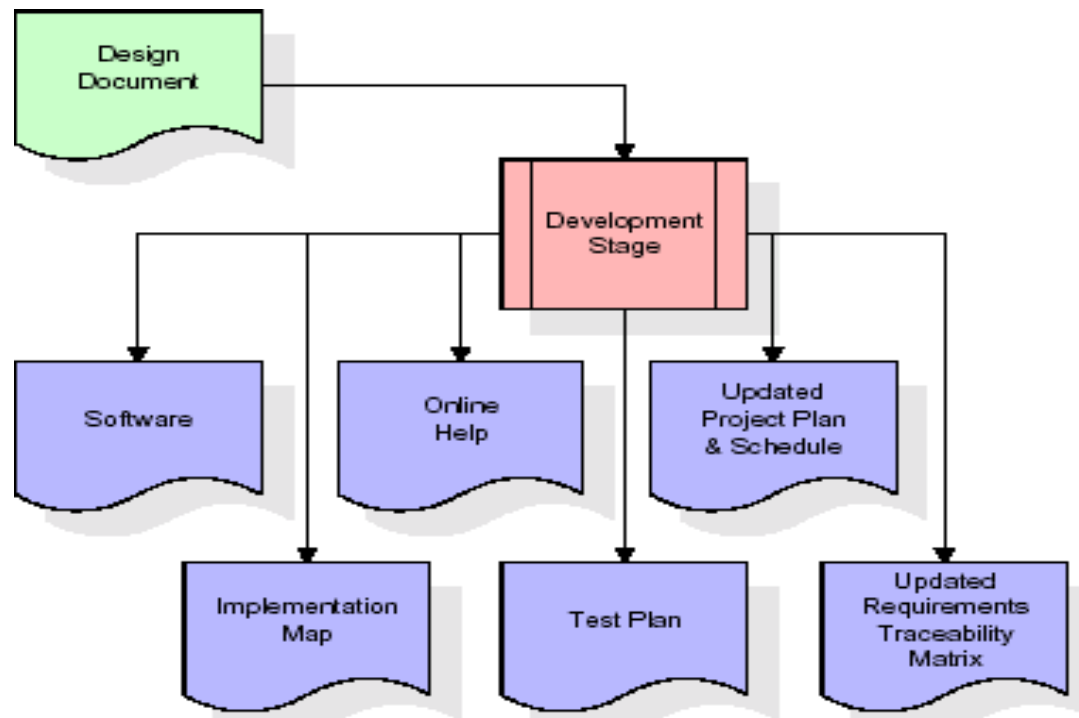
The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.



When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

Development (Coding) Stage:

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.

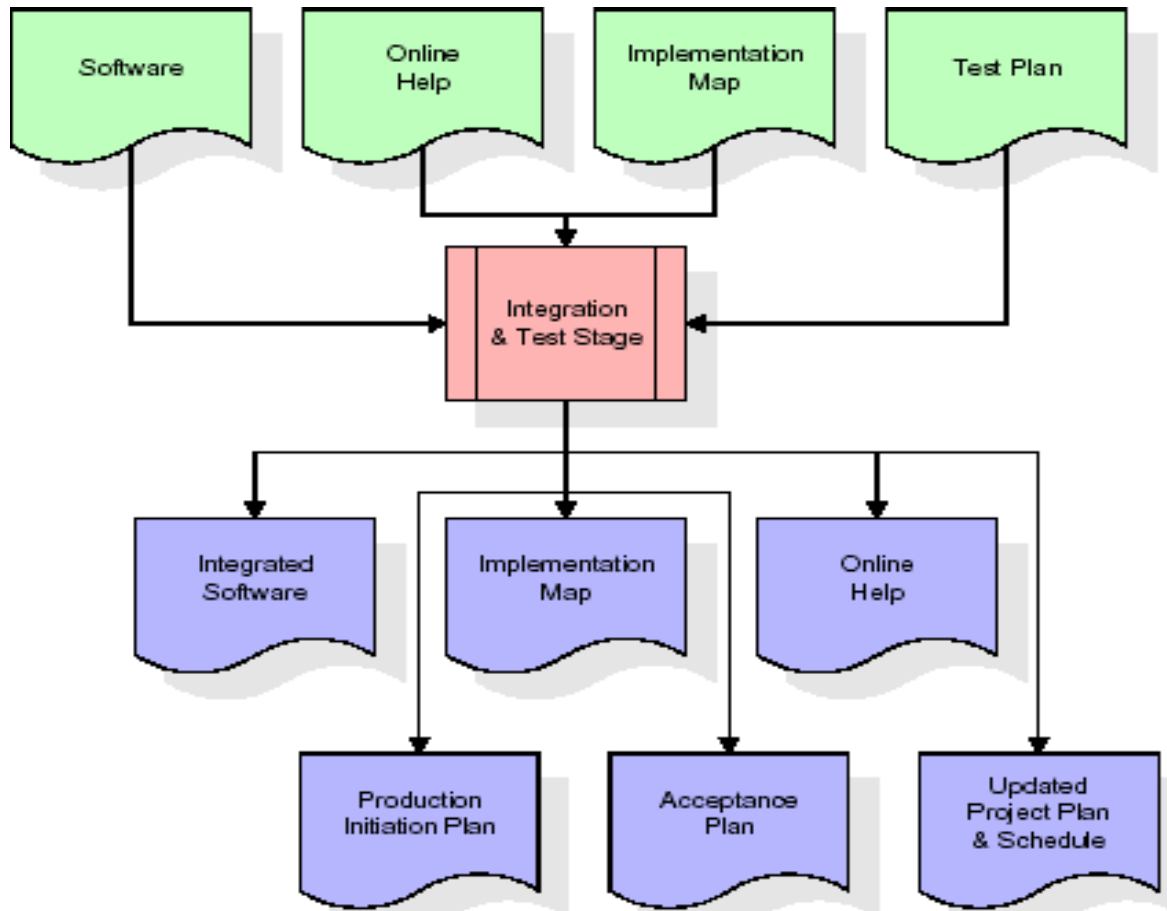


The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

Integration & Test Stage:

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and

production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.

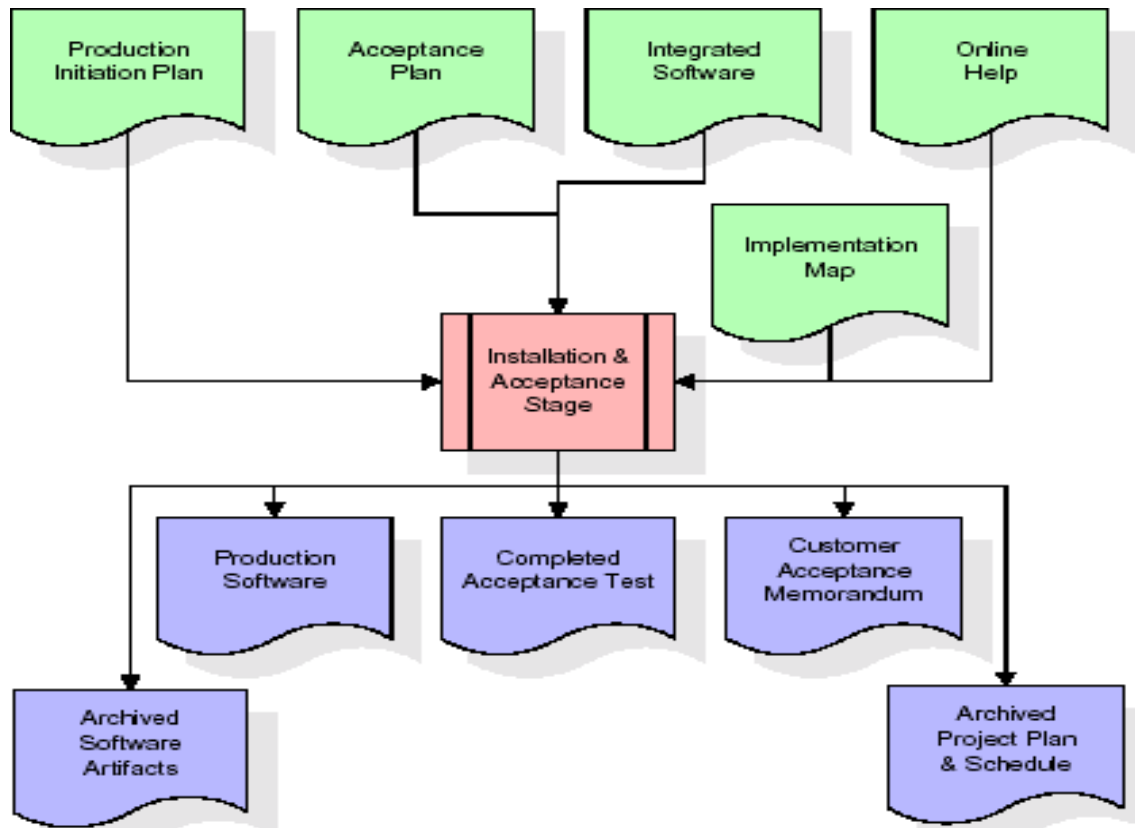


The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

- **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

Maintenance:

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

3.5 Software Requirement Specification

Overall Description

A Software Requirements Specification (SRS) – a requirements specification for a software system is a complete description of the behavior of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Nonfunctional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the systems development lifecycle domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

1. Business requirements describe in business terms what must be delivered or accomplished to provide value. Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.) Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resource and infinite time. There are aspects in the feasibility study portion of the preliminary investigation

ECONOMIC FEASIBILITY

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

OPERATIONAL FEASIBILITY

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

TECHNICAL FEASIBILITY

Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to .the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

3.6 External Interface Requirements

User Interface

The user interface of this system is a user friendly python Graphical User Interface.

Hardware Interfaces

The interaction between the user and the console is achieved through python capabilities.

Software Interfaces

The required software is python. **Operating Environment** Windows XP.

HARDWARE REQUIREMENTS:

- | | |
|------------|-----------------------------|
| •Processor | - Pentium –IV |
| •Speed | - 1.1 Ghz |
| •RAM | - 256 MB(min) |
| •Hard Disk | - 20 GB |
| •Key Board | - Standard Windows Keyboard |
| •Mouse | - Two or Three Button Mouse |
| •Monitor | - SVGA |

SOFTWARE REQUIREMENTS:

- | | |
|-----------------------|---------------|
| •Operating System | - Windows7/8/ |
| •Programming Language | - Python |

CHAPTER-4

SYSTEM DESIGN

UML diagrams:

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

- **User Model View**

- i. This view represents the system from the user's perspective.
- ii. The analysis representation describes a usage scenario from the end-users perspective.

- **Structural Model view**

- i. In this model the data and functionality are arrived from inside the system.
- ii. This model view models the static structures.

- **Behavioral Model View**

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

- **Implementation Model View**

In this the structural and behavioral as parts of the system are represented as they are to be built.

- **Environmental Model View**

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

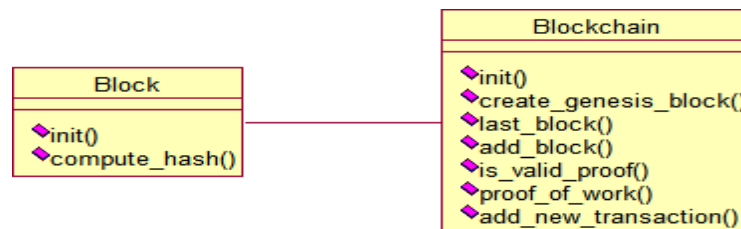
4.1 CLASS DIAGRAM:-

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the

application and the classes to be programmed. A class with three sections, in the diagram, classes is represented with boxes which contain three parts:

- The upper part holds the name of the class
- The middle part contains the attributes of the class
- The bottom part gives the methods or operations the class can take or undertake

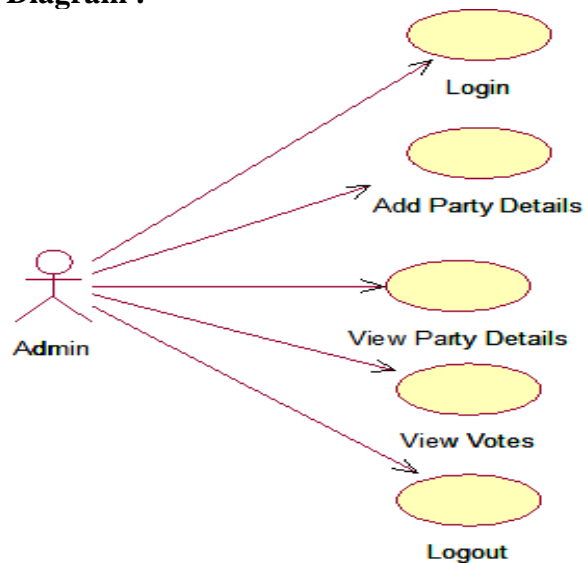
Class diagram:



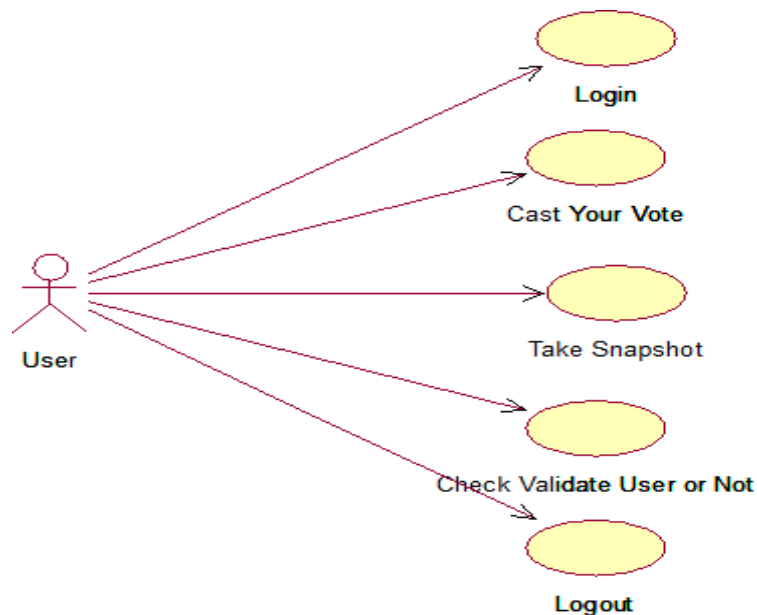
4.2 USE CASE DIAGRAM:-

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

Use case for admin Diagram :



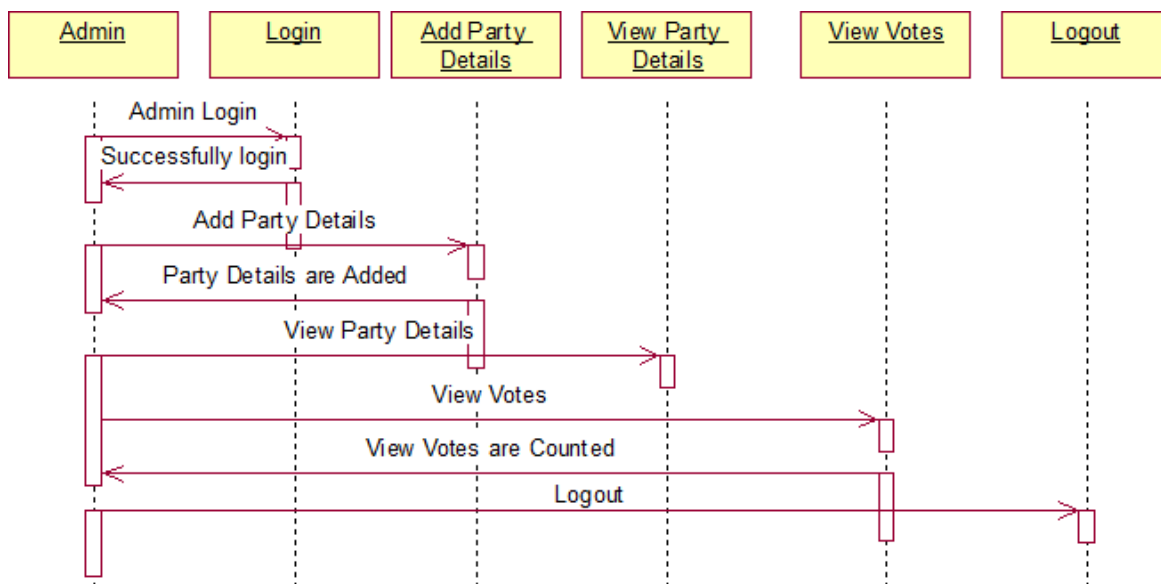
Use Case diagram for User:



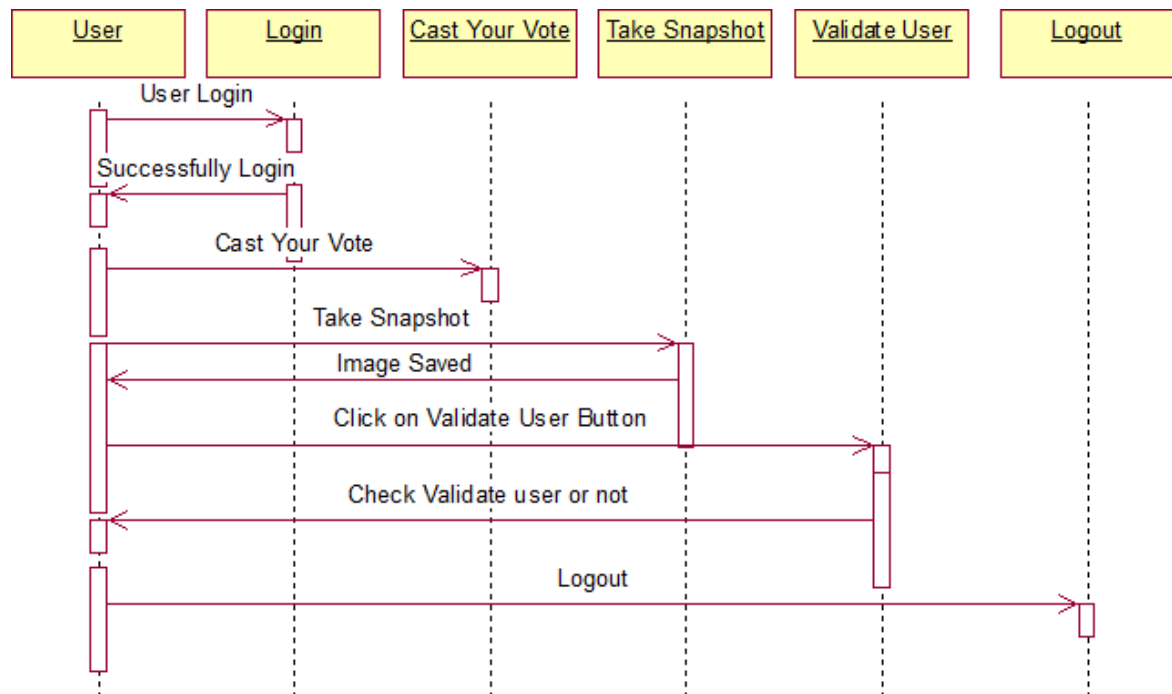
4.3. SEQUENCE DIAGRAM:

A **sequence diagram** is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams**, **event scenarios**, and timing diagrams.

Sequence diagram for Admin:



Sequence diagram for User:

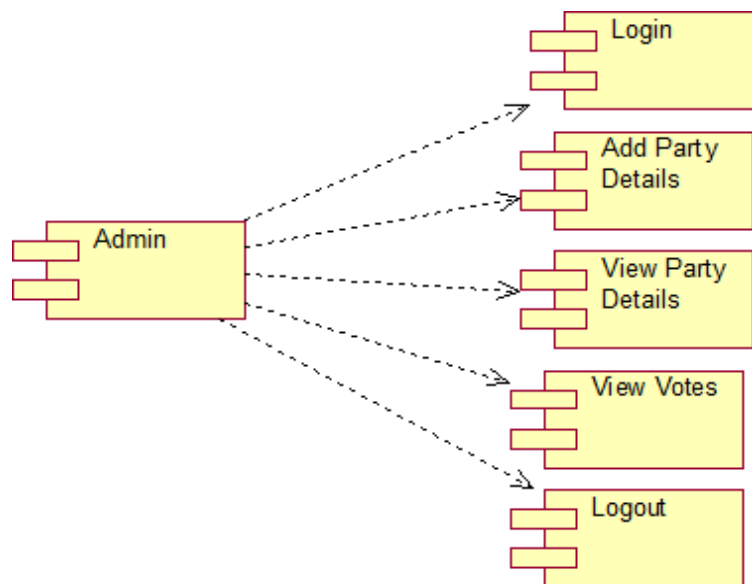


4.4 COMPONENT DIAGRAM

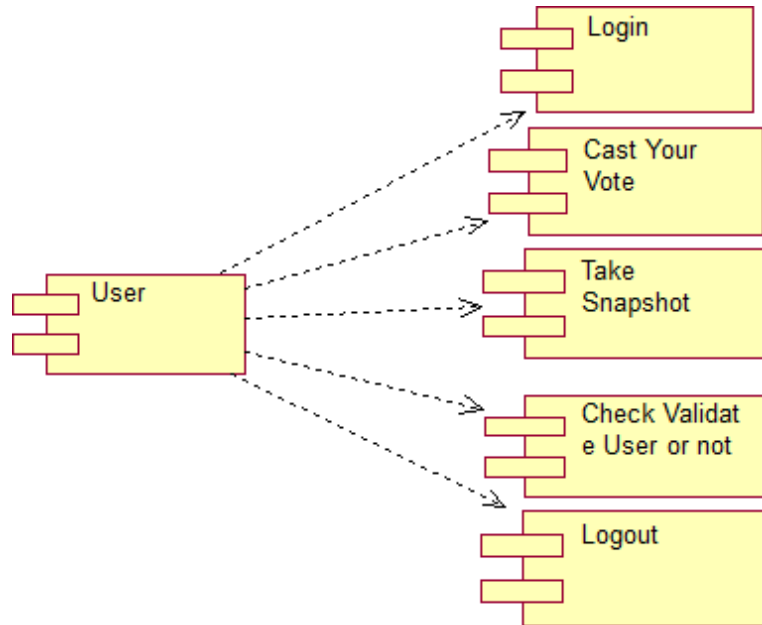
In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

Component diagram for Admin:



Component diagram for User:

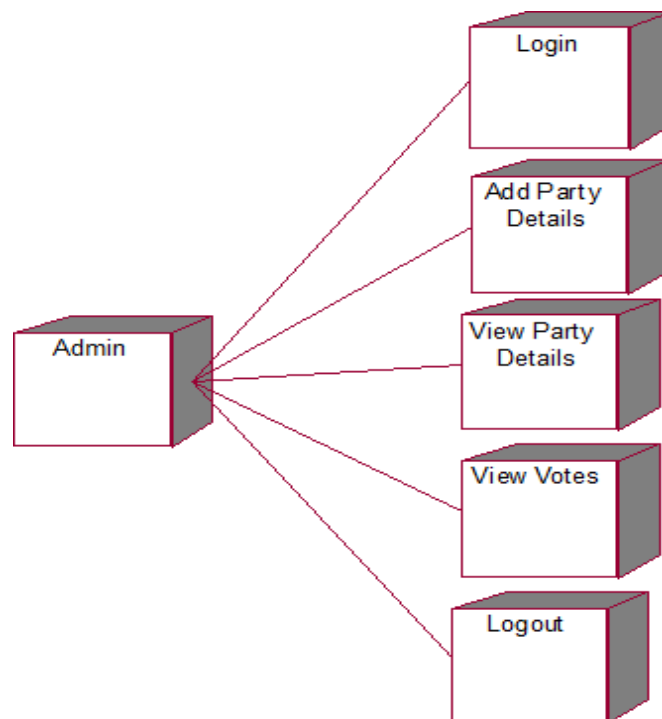


4.5 DEPLOYMENT DIAGRAM

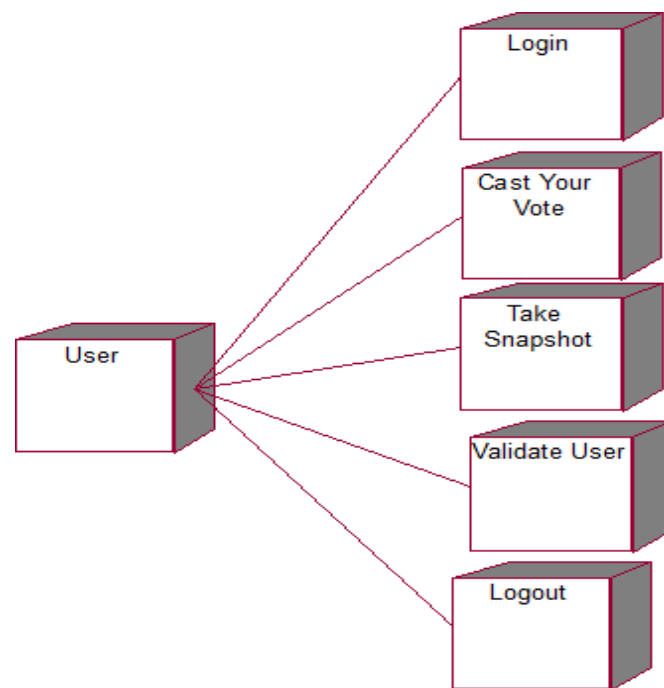
A **deployment diagram** in the Unified Modeling Language models the physical deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

Deployment diagram for Admin:



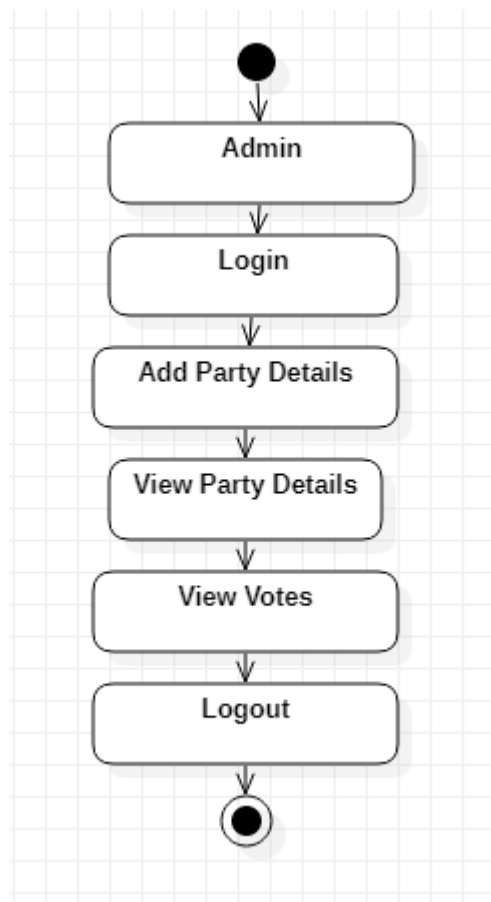
Deployment diagram for User:



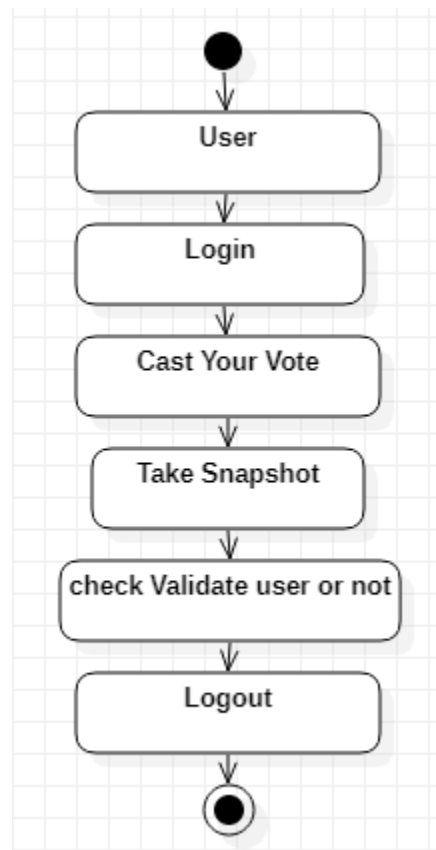
4.6 ACTIVITY DIAGRAM:

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.

Activity diagram for Admin:



Activity diagram for user:

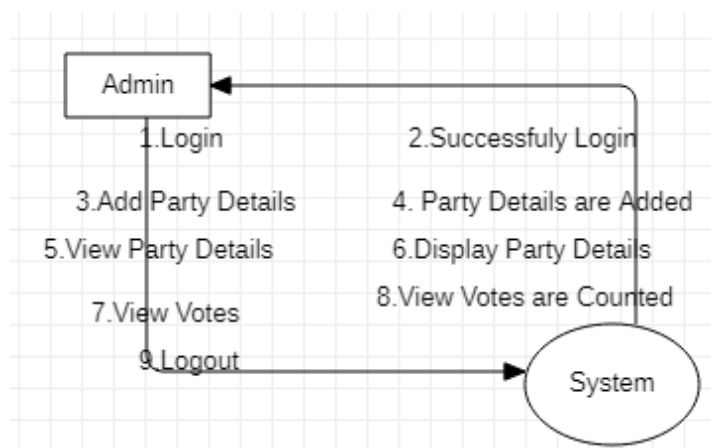


4.7 DATA FLOW DIAGRAM :

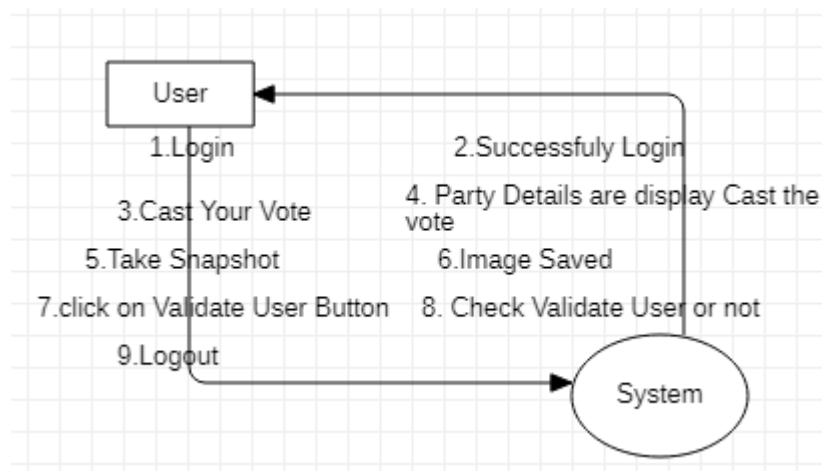
Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

Data Flow Diagram for Admin:



Data Flow Diagram for User:



CHAPTER-5

IMPLEMENTATION

5.1PYTHON

Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

History of Python:

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

Why Python was created?

In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python.

Why the name Python?

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late seventies. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

Features of Python:

A simple language which is easier to learn

Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax.

If you are a newbie, it's a great choice to start your journey with Python.

Free and open-source

You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software's written in it, you can even make changes to the Python's source code. Python has a large community constantly improving it in each iteration.

Portability

You can move Python programs from one platform to another, and run it without any changes. It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

Extensible and Embeddable

Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code.

This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

A high-level, interpreted language

Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on.

Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.

Large standard libraries to solve common tasks

Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server? You can use MySQLdb library using import MySQLdb .

Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.

Object-oriented

Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.

With OOP, you are able to divide these complex problems into smaller sets by creating objects.

Applications of Python:

1. Simple Elegant Syntax

Programming in Python is fun. It's easier to understand and write Python code. Why? The syntax feels natural. Take this source code for an example:

```
a = 2
```

```
b = 3
```

```
sum = a + b
```

```
print(sum)
```

2. Not overly strict

You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement.

Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

3. Expressiveness of the language

Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of Tic-tac-toe game with a graphical interface and a smart computer opponent in less than 500 lines of code. This is just an example. You will be amazed how much you can do with Python once you learn the basics.

4. Great Community and Support

Python has a large supporting community. There are numerous active forums online which can be handy if you are stuck.

5.2SAMPLE CODE:

Block.py

```
from hashlib import sha256 import json

import time class Block:

def __init__(self, index, transactions, timestamp, previous_hash): self.index = index

self.transactions = transactions self.timestamp = timestamp self.previous_hash =

previous_hash self.nonce = 0

def compute_hash(self):
"""
A function that return the hash of the block contents. """
block_string = json.dumps(self.__dict__, sort_keys=True) return
sha256(block_string.encode()).hexdigest()
```

Blockchain.py

```
from hashlib import sha256 import json
```

```
import time import pickle
```

```
from datetime import datetime import random
```

```
import pyaes, pbkdf2, binascii, os, secrets import base64
```

```
class Block:
```

```
    def __init__(self, index, transactions, timestamp, previous_hash): self.index = index
```

```
    self.transactions = transactions self.timestamp = timestamp self.previous_hash =
```

```
        previous_hash
```

```
    self.nonce = 0
```

```
    def compute_hash(self):
```

```
        block_string = json.dumps(self.__dict__, sort_keys=True) return
```

```
        sha256(block_string.encode()).hexdigest()
```

```
class Blockchain:
```

```
    # difficulty of our PoW algorithm
```

```
    difficulty = 2 #using difficulty 2 computation def __init__(self):
```

```
        self.unconfirmed_transactions = [] self.chain = [] self.create_genesis_block() self.peer
```

```
        = []
```

```
        self.translist = []
```

```
    def create_genesis_block(self): #create genesis block genesis_block = Block(0, [],
```

```

time.time(), "0") genesis_block.hash = genesis_block.compute_hash()

self.chain.append(genesis_block)

@property

def last_block(self): return self.chain[-1]

def add_block(self, block, proof): #adding data to block by computing new and
previous hashes

previous_hash = self.last_block.hash

if previous_hash != block.previous_hash: return False

if not self.is_valid_proof(block, proof): return False

block.hash = proof #print("main "+str(block.hash)) self.chain.append(block)

return True

def is_valid_proof(self, block, block_hash): #proof of work

return (block_hash.startswith('0'* Blockchain.difficulty)and
block_hash == block.compute_hash())

def proof_of_work(self, block): #proof of work block.nonce = 0

computed_hash = block.compute_hash()

while not computed_hash.startswith('0' * Blockchain.difficulty): block.nonce += 1

computed_hash = block.compute_hash() def add_new_transaction(self, transaction):

self.unconfirmed_transactions.append(transaction) def addPeer(self, peer_details):

self.peer.append(peer_details)

def addTransaction(self,trans_details): #add transaction

```

```
self.translist.append(trans_details)

def mine(self):#mine transaction

if not self.unconfirmed_transactions: return False

last_block = self.last_block

new_block = Block(index=last_block.index + 1,

transactions=self.unconfirmed_transactions, timestamp=time.time(),

previous_hash=last_block.hash)

proof = self.proof_of_work(new_block) self.add_block(new_block, proof)

self.unconfirmed_transactions = [] return new_block.index

def save_object(self,obj, filename): with open(filename, 'wb') as output:

pickle.dump(obj, output, pickle.HIGHEST_PROTOCOL)
```

CHAPTER – 6

TESTING

Implementation and Testing:

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

Implementation

The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

Module Testing

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

Integration Testing

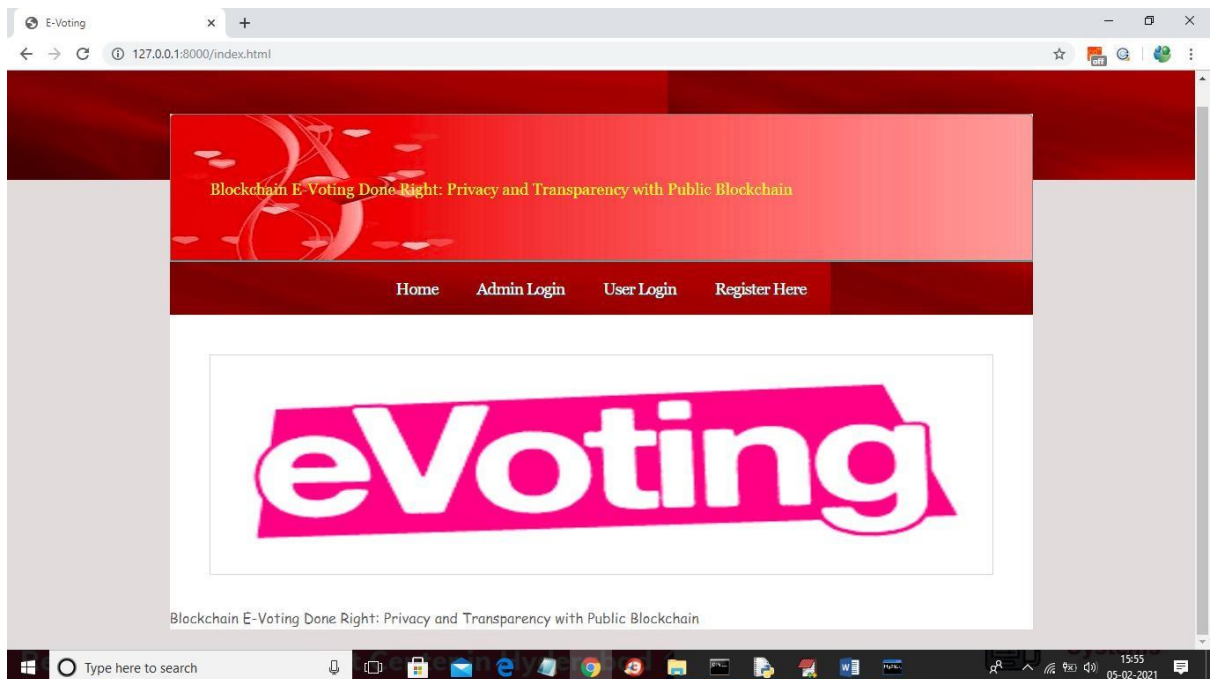
After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

Acceptance Testing :

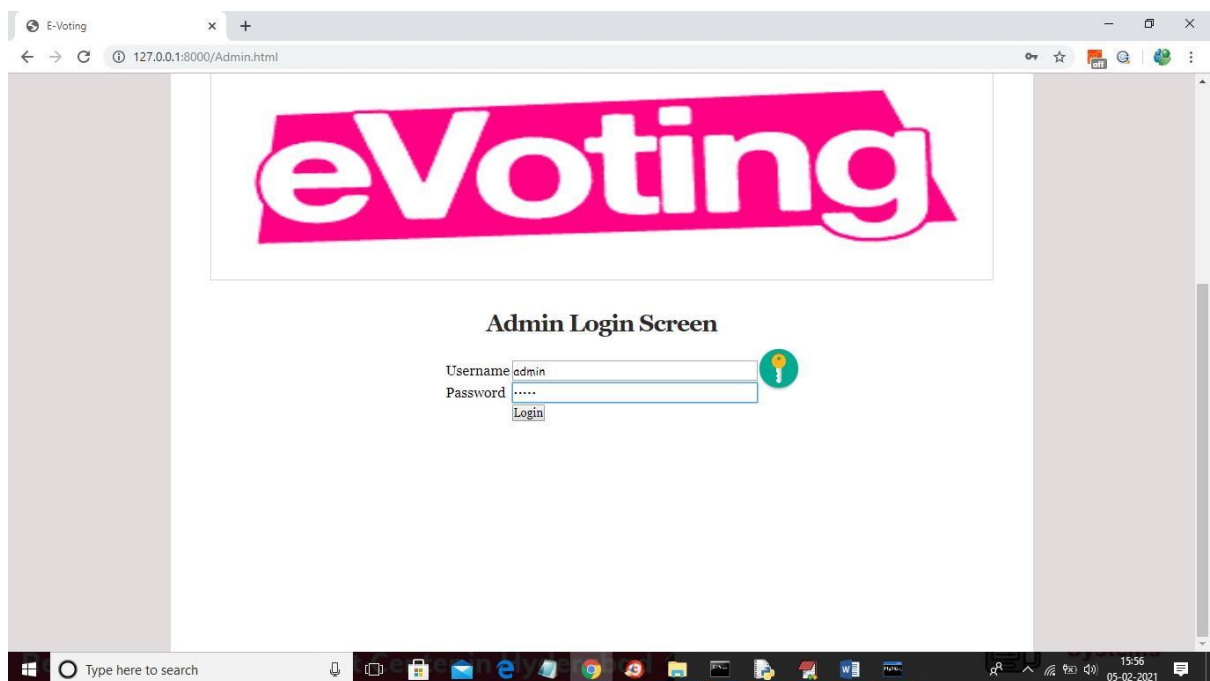
When that user find no major problems with its accuracy, the system passes through a final acceptance test. This test confirms that the system meets the original goals, objectives and requirements established during analysis without actual execution which eliminates wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

Test Case Id	Test Case Name	Test Case Desc.	Test Steps			Test Case Status	Test Priority
			Step	Expected	Actual		
01	Admin Login	Verify the Admin login or not	If the Admin may not login	Admin cannot Add Party Details	Admin can Add Party Details	High	High
02	User Login	Verify the User login or not	If the User may not login	User cannot Cast Your Vote	User can Cast Your Vote	High	High

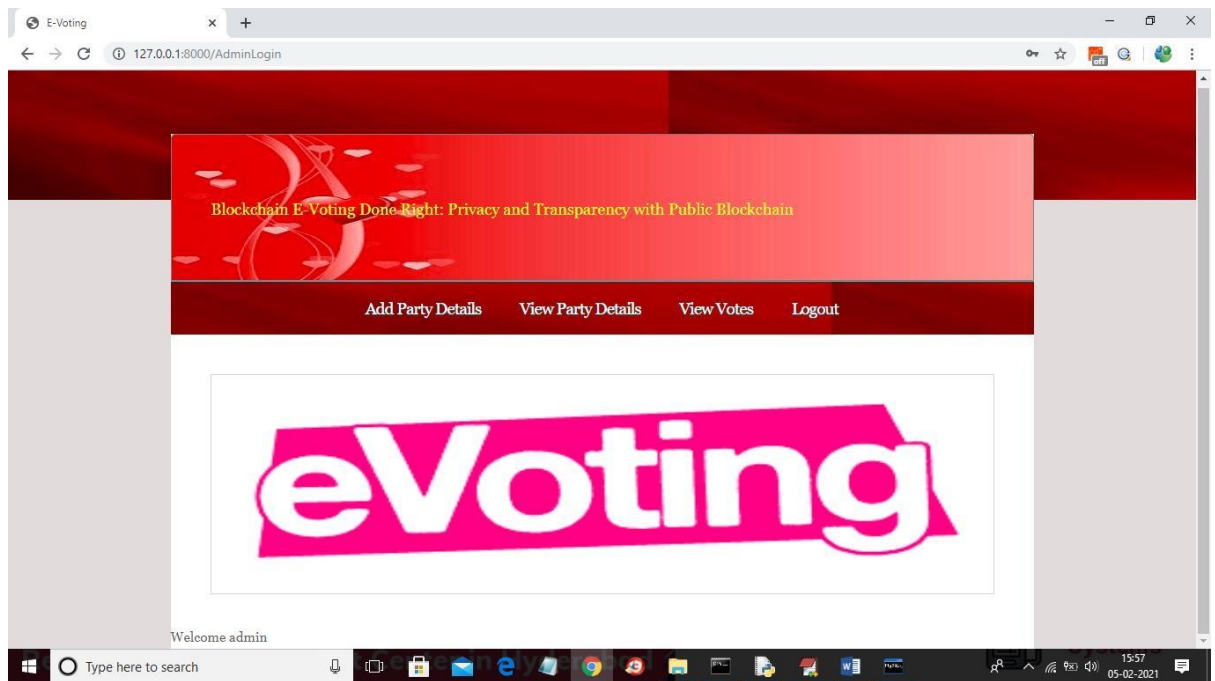
CHAPTER-7 SCREENSHOTS



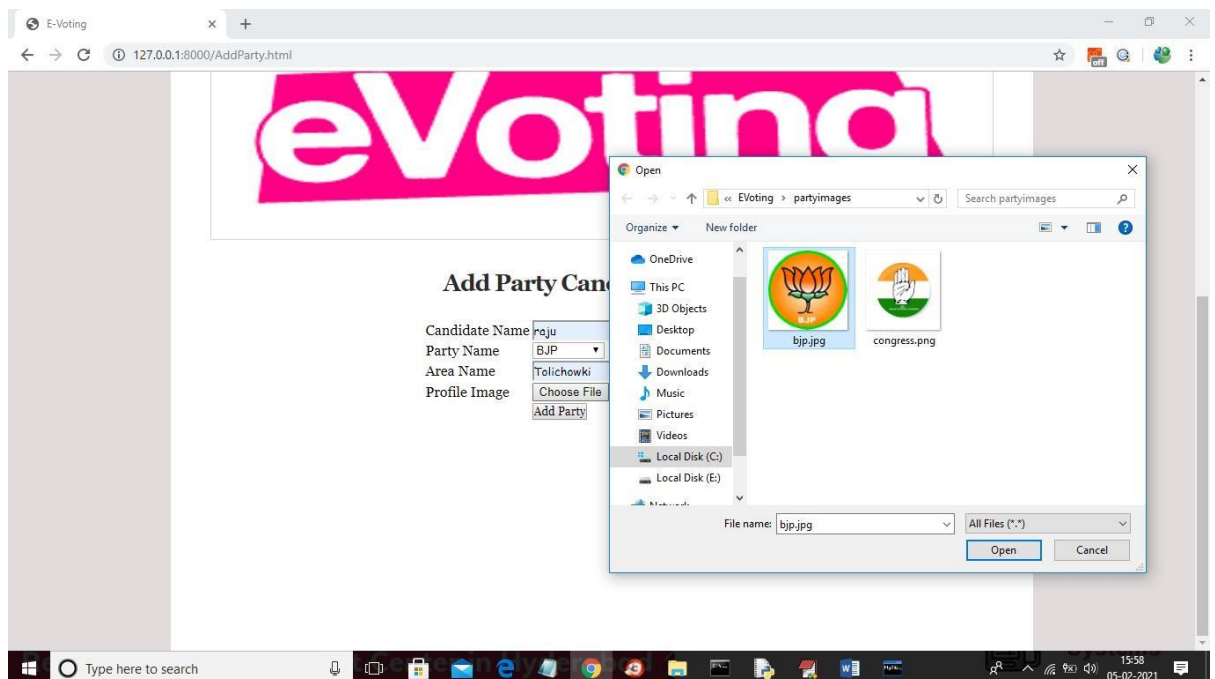
In above screen click on 'Admin Login' link to get below screen



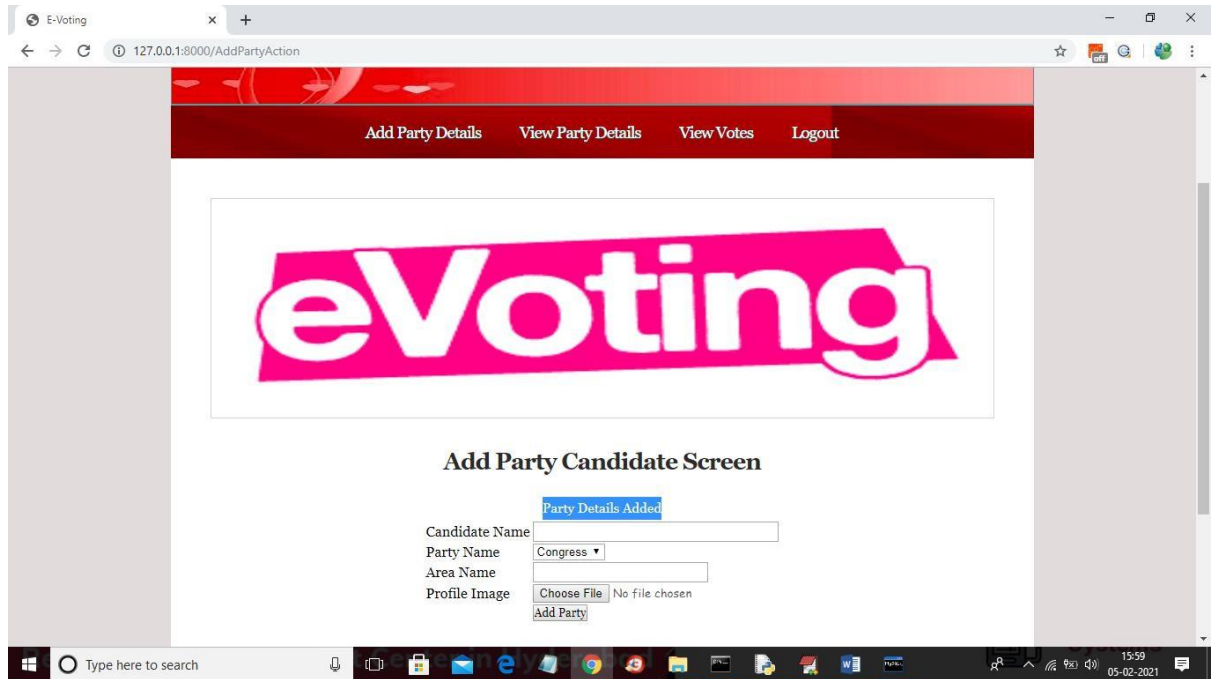
In above screen login as admin by giving username as 'admin' and password as 'admin' and then click Login button to get below screen



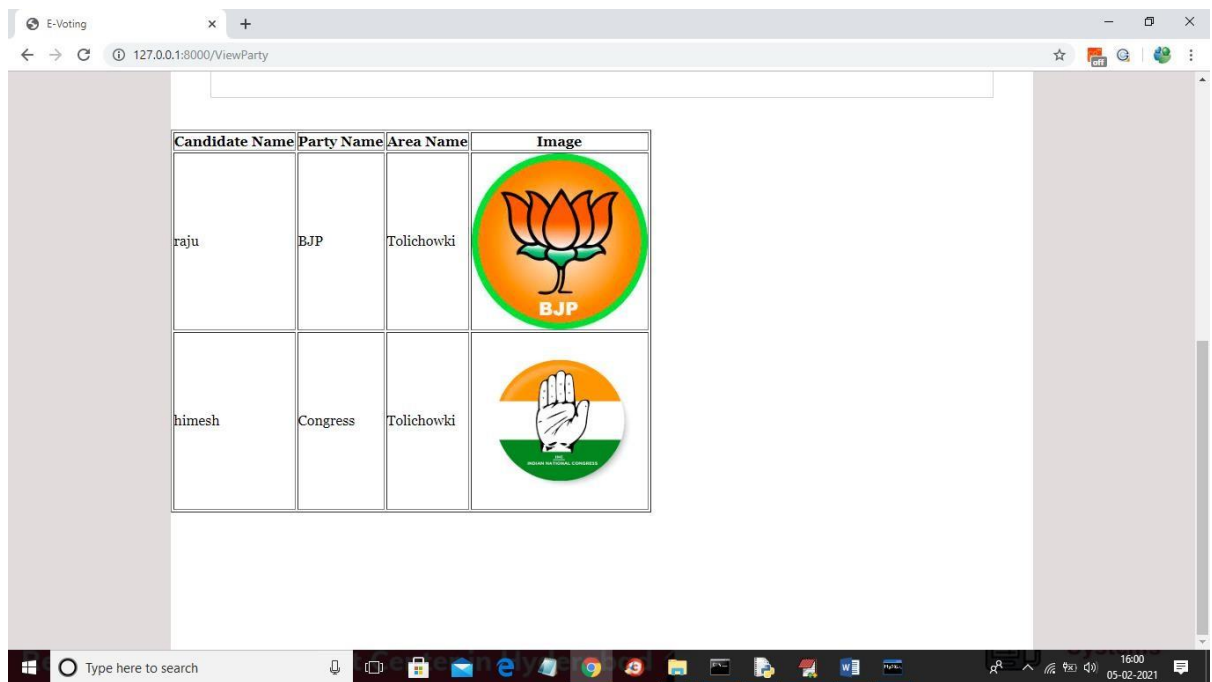
In above screen admin can click on 'Add Party Details' link to add party details



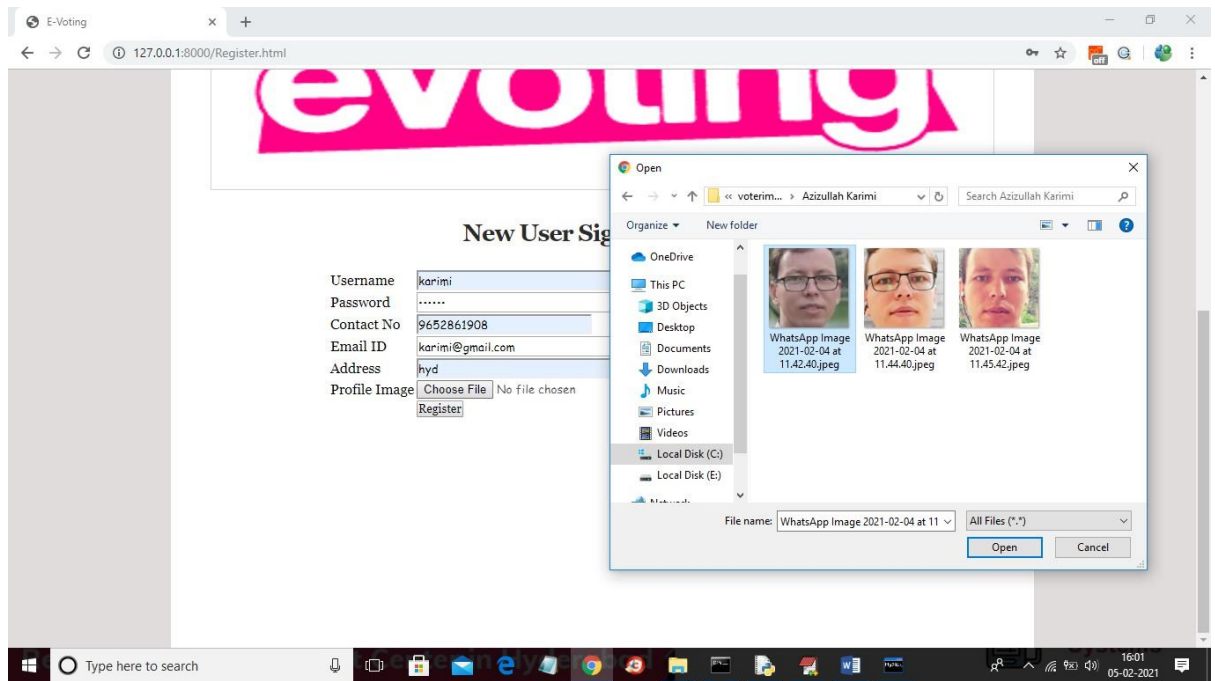
In above screen adding party and candidate details and then upload image and click on 'Open' button then click on 'Add Party' button to add party details



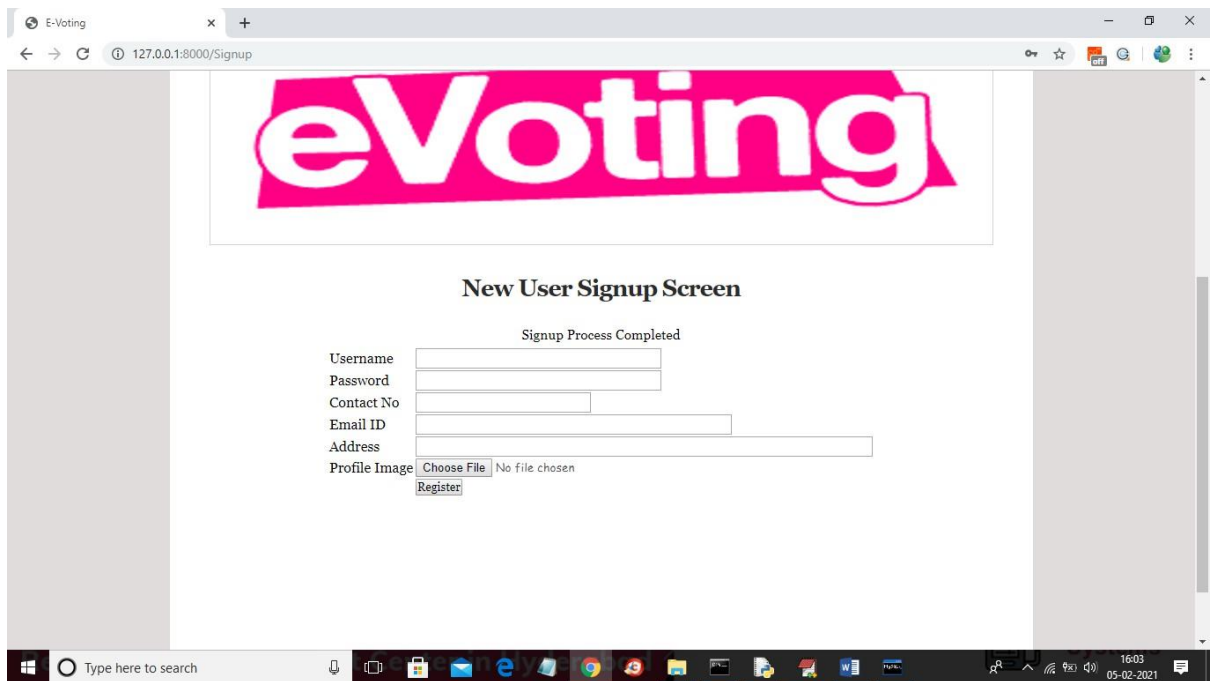
In above screen part details added and similarly you can add any number of party members and now click on 'View Party Details' link to get below screen



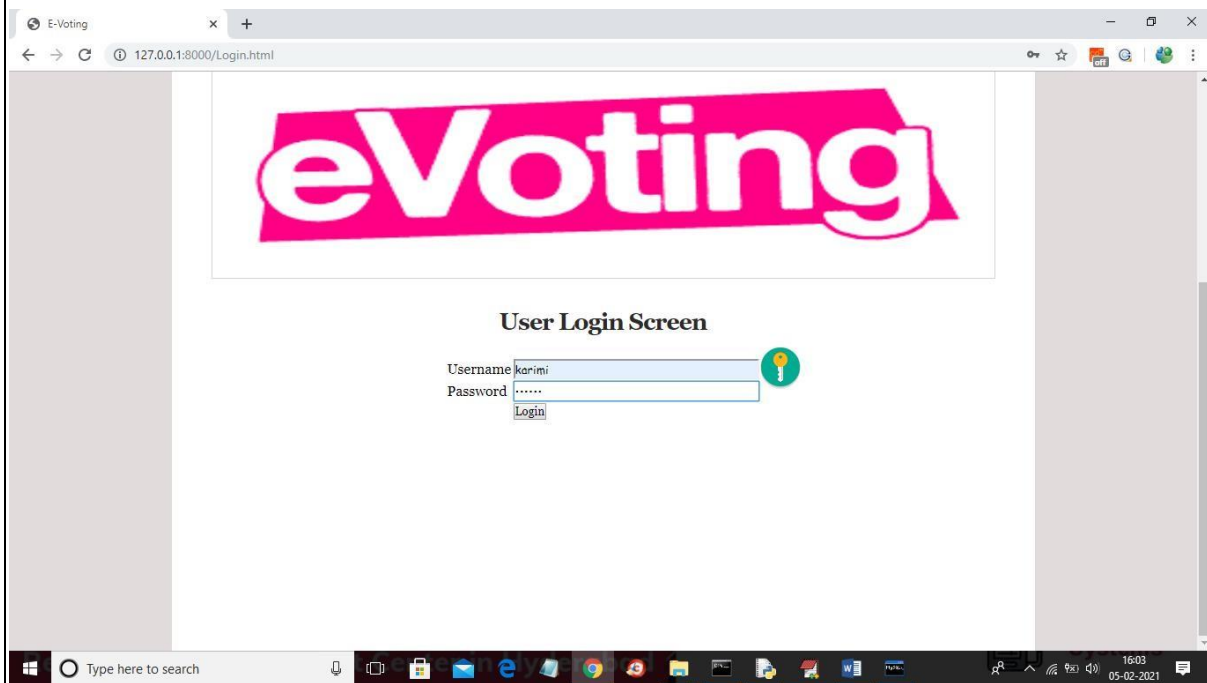
In above screen displaying add added party details and now click on 'Logout' link to logout as admin and then add new user



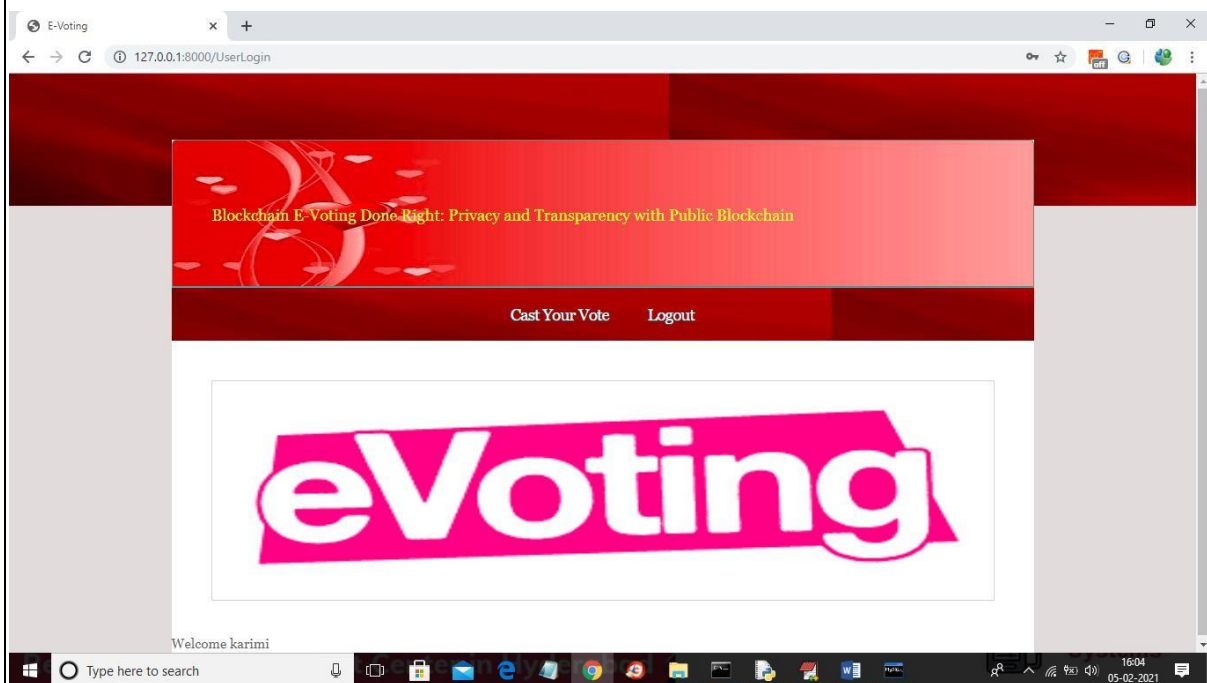
In above screen adding new user and then selecting his face photo taken from webcam and then click on 'Register' button to complete signup process. Here you have given images taken from phone but we need to capture from webcam for dataset as quality of webcam image and phone image vary and then problem comes in prediction.



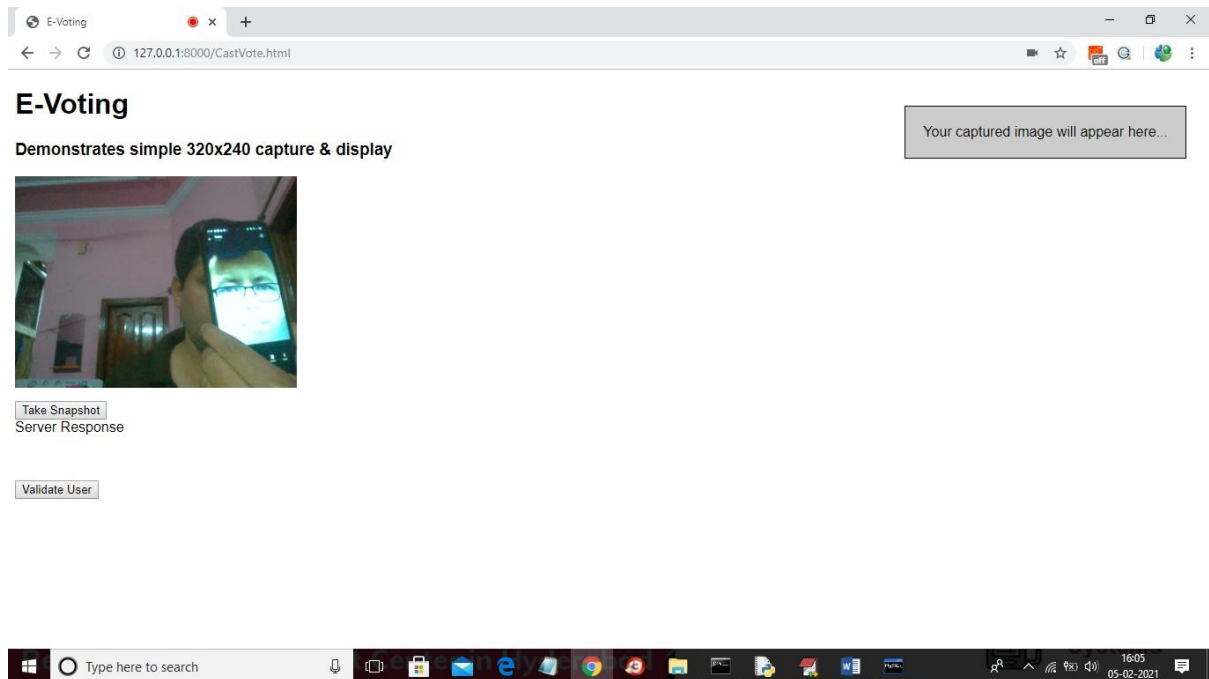
In above screen signup process completed and now login as this user to cast vote



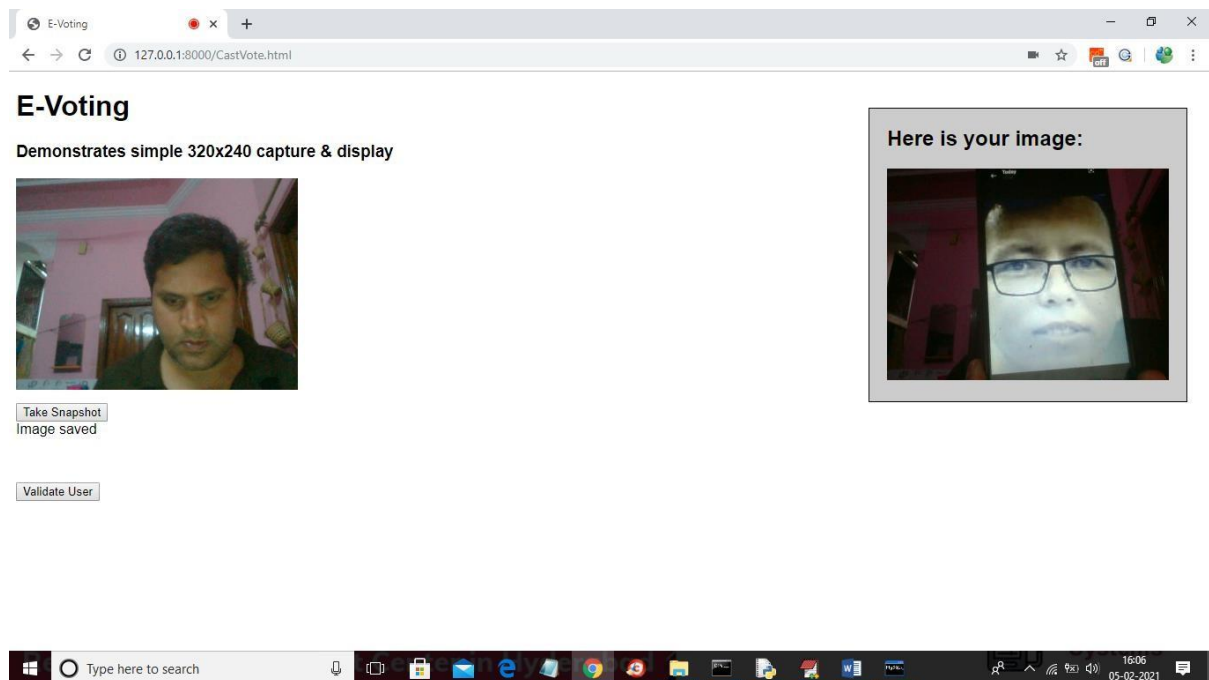
In above screen application first authenticate user by using his login details and once after successful login then user will get below page



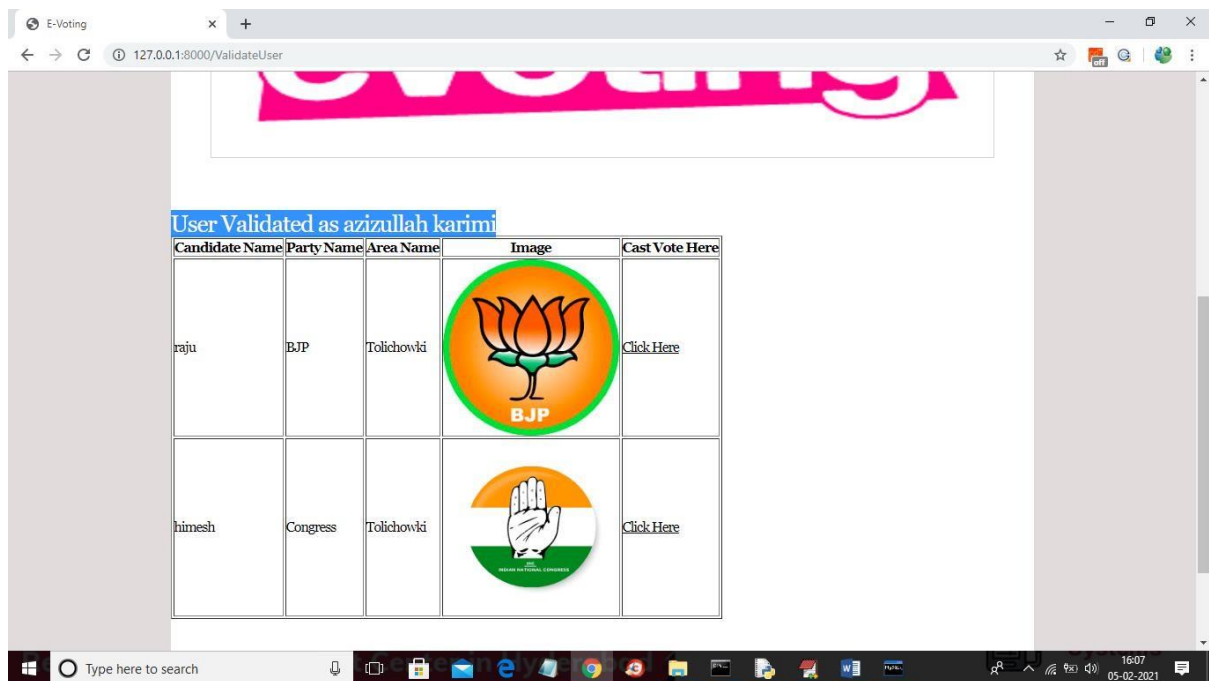
In above screen user can click on 'Cast Your Vote' link to get below webcam screen



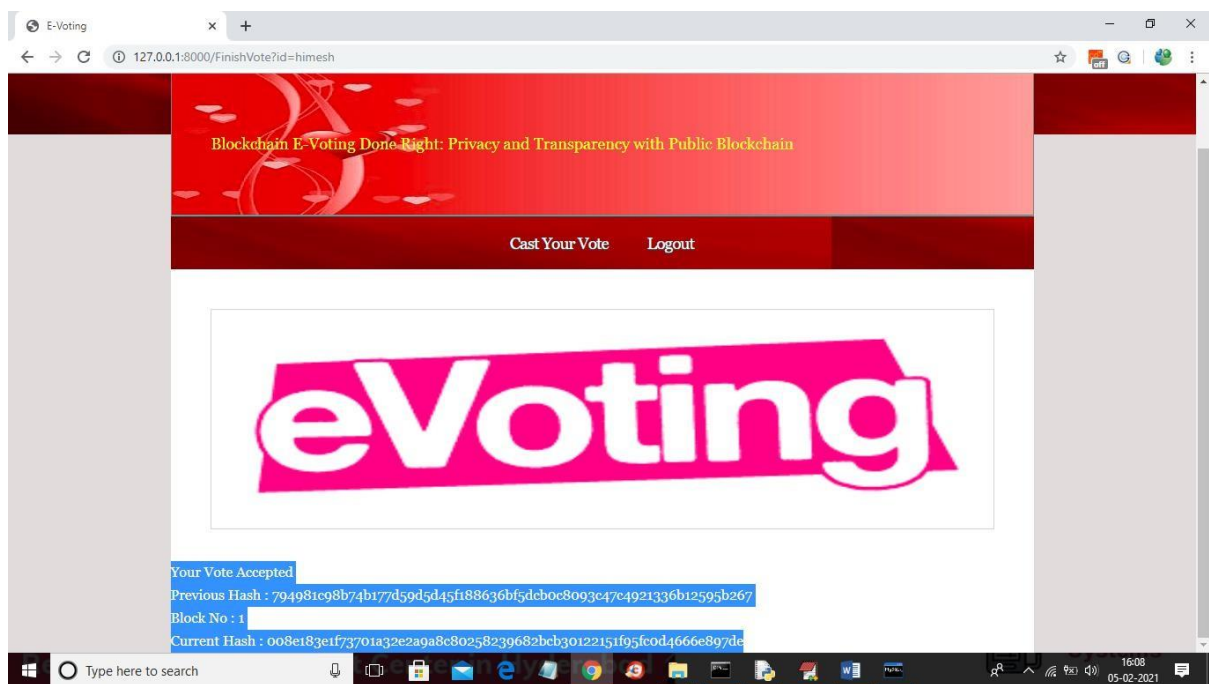
In above screen webcam is running and then by showing person face we need to click on 'Take Snapshot' button to capture his face



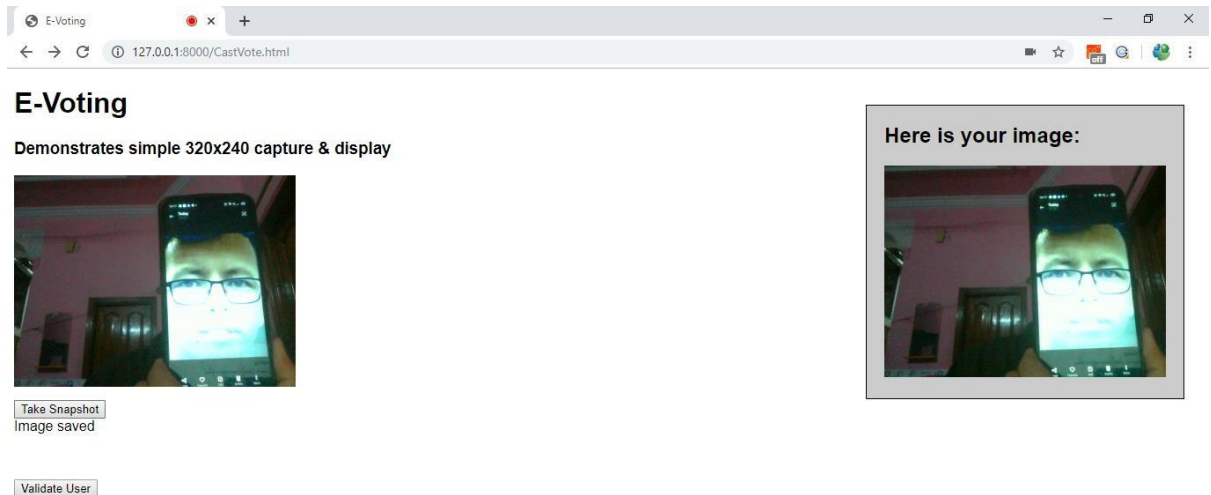
In above screen person faces is capture and now click on 'Validate User' button to validate user



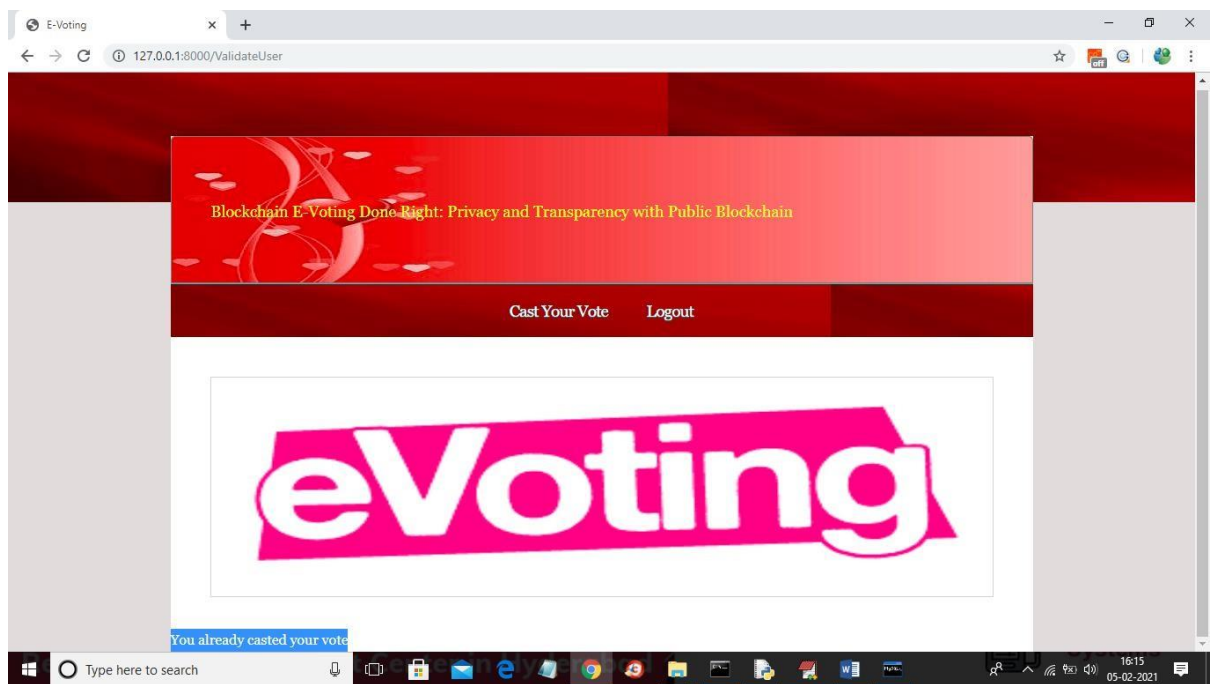
In above screen in blue colour you can see user is identified as ‘azizullahkarimi’ and then displaying list of candidates and now user can click on ‘Click Here’ option to cast his vote and to get below screen



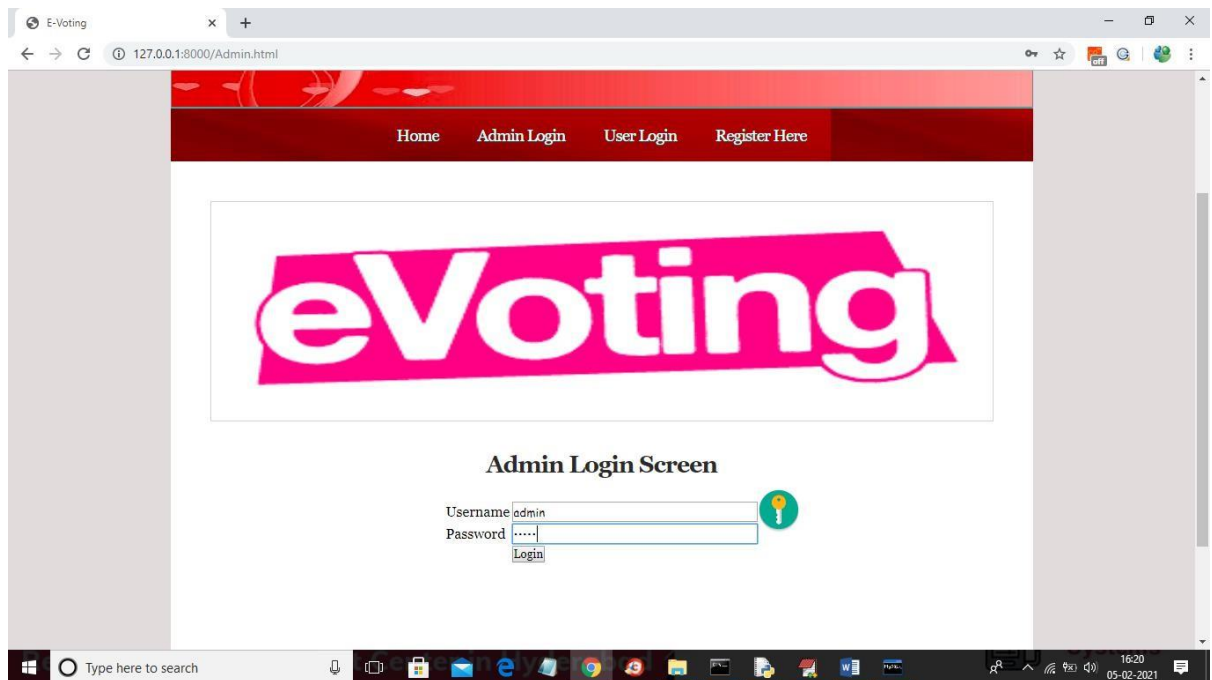
In above screen as this is the first vote so block will be added to Blockchain with block No as 1 and we can see Blockchain created a chain of blocks with previous and current hash



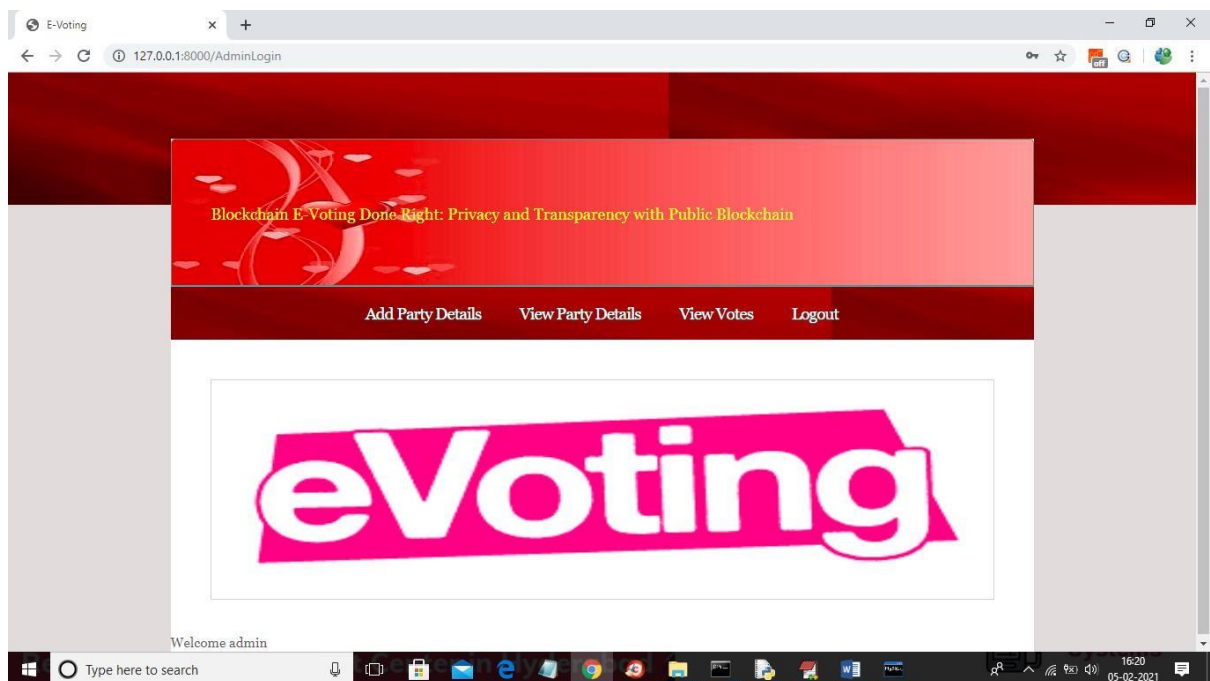
In above screen same user trying again and below is the result



In above screen if same user try again then will get message as 'You already casted you vote' and now logout and login as 'admin' to get vote count





In above screen login as admin and after login will get below screen



In above screen admin can click on 'View Votes' link to get below screen

E-Voting x +

127.0.0.1:8000/ViewVotes

Candidate Name	Party Name	Area Name	Image	Vote Count
raju	BJP	Tolichowki		0
himesh	Congress	Tolichowki		1

Type here to search

16:22 05-02-2021

In above screen admin can view all vote counts.

CHAPTER-8

CONCLUSION :

Although we can see slight differences in network times, they are so negligible that public blockchain has more advantages in such an electoral system due to its openness of data and that anyone can watch them in the real time. A private blockchain is a bit faster, but it reduces the credibility of the whole system by being partially centralized because it only runs where the authority wants it. The table shows that the average times to add one person's voice are: Ganache 6.32 s (median 6.34 s), Hyper ledger Composer 6.05 s (median 6.04 s), and Ethereum Ropsten 17.75 s (median 17.93 s). These times are influenced by the used consensus algorithm and also by the block time

CHAPTER -9

REFERENCES

- [1] N. Kshetri and J. Voas, "Blockchain-Enabled E-Voting," IEEE Software, vol. 35, pp. 95-99, jul 2018.
- [2] M. Pawlak, J. Guziur, and A. Poniszewska-Maranda, "Voting Process with Blockchain Technology: Auditable Blockchain Voting System," in Lecture Notes on Data Engineering and Communications Technologies, pp. 233-244, Springer, Cham, 2019.
- [3] B. Singhal, G. Dhameja, and P. S. Panda, "How Blockchain Works," in Beginning Blockchain, pp. 31-148, Berkeley, CA: Apress, 2018.
- [4] Agora, "Agora Whitepaper," 2018.
- [5] R. Perper, "Sierra Leone is the first country to use blockchain during an election - Business Insider," 2018.
- [6] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," tech.rep., 2008.
- [7] G. Wood et al., "Ethereum: A secure decentralized generalized transaction ledger," Ethereum project yellow paper, vol. 151, pp. 1-32, 2014.
- [8] S. Landers, "Netvote: A Decentralized Voting Platform – Netvote Project Medium," 2018.
- [9] P. McCorry, S. F. Shahandashti, and F. Hao, "A Smart Contract for Boardroom Voting with Maximum Voter Privacy," in Lecture Notes in Computer Science, ch. FCDS, pp. 357-375, Springer, Cham, 2017.
- [10] Z. Brakerski and V. Vaikuntanathan, "Efficient Fully Homomorphic Encryption from (Standard) LWE," SIAM Journal on Computing, vol. 43, pp. 831-871, jan 2014.
- [11] O. Goldreich and Y. Oren, "Definitions and properties of zero knowledge proof systems," Journal of Cryptology, vol. 7, no. 1, pp. 1-32, 1994.