**INTRODUCTION**

“ONLINE VOTING SYSTEM” is an online voting technique. In this system people who have citizenship of India and whose age is above 18 years of age and any sex can give his\her vote online without going to any physical polling station There is a database which is maintained in which all the names of voters with complete information is stored.

In “ONLINE VOTING SYSTEM” a voter can use his\her voting right online without any difficulty. He\She has to be registered first for him/her to vote. Registration is mainly done by the system administrator for security reasons. The system Administrator registers the voters on a special site of the system visited by him only by simply filling a registration form to register voter. Citizens seeking registration are expected to contact the system administrator to submit their

details. After the validity of them being citizens of India has been confirmed by the system administrator by comparing their details submitted with those in existing databases such as those as the Registrar of Persons, the citizen is then registered as a voter.

After registration, the voter is assigned a secret Voter ID with which he/she can use to log into the system and enjoy services provided by the system such as voting. If invalid/wrong details are submitted, then the citizen is not registered.

**1.1 AIM**

**The main purposes of OVS include:**

**1.1.1 Provision of improved voting services to the voters through fast, timely and convenient voting.**

**1.1.2 Reduction of the costs incurred by the Indian Electoral Commission**

**during voting time in paying the very many clerks employed for the sake of**

**the success of the manual system**.

**1.1.3 Check to ensure that the members who are registered are the only ones to vote. Cases of “Dead People” voting are also minimized.**

**1.1.4 Online voting system (OVS) will require being very precise or cost cutting to produce an effective election management system.**

**1.2 OBJECTIVE**

**The specific objectives of the project include:**

**1.2.1 Reviewing the existing/current voting process or approach in India;**

**1.2.2 Coming up with an automated voting system in India;**

**1.2.3 Implementing an automated/online voting system;**

**1.2.4 Validating system to ensure that only legible voters are allowed to vote.**

**1.2.5 Many users like army persons or NRI cannot come to the voting place. Therefore we have to implement an online voting system by which the voter can vote over the online.**

**1.3 SCOPE**

It is focused on studying the existing system of voting in India and to

make sure that the peoples vote is counts, for fairness in the elective

positions. This is also will produce:

Less effort and less labor intensive, as the primary cost and focus primary on creating, managing, and running a secure web voting portal.

Increasing number of voters as individuals will find it easier and more convenient to vote, especially those abroad.

**1.4 EXISTING SYSTEM**

**1.4.1** **Paper-based voting**: The voter gets a blank ballot and use a pen or a

marker to indicate he want to vote for which candidate. Hand-counted

ballots is a time and labor consuming process, but it is easy to

manufacture paper ballots and the ballots can be retained for

verifying, this type is still the most common way to vote.

**1.4.2** **Punch card**: The voter uses metallic hole-punch to punch a hole on the

blank ballot. It can count votes automatically, but if the voter’s

perforation is incomplete, the result is probably determined wrongfully.

**1.4.3**.**Direct recording electronic voting machine**: This type, which is

abbreviated to DRE, integrates with keyboard; touch screen, or buttons

for the voter press to poll. Some of them lay in voting records and

counting the votes is very quickly. But the other DRE without keep

voting records are doubted about its accuracy

**1.5 PROPOSED SYSTEM**

The project maintain two level of users:-

**1.5.1** Administrator level

**1.5.2** Voter level

Main facilities available in this project are:-

* Maintaining voter’s identification.
* Providing online voting management.
* Providing updation of voter’s and candidate’s information.
* Providing voter information and candidate’s information to election commission of India.
* Voter can give his\her vote from any part of India.

**2. PROJECT ANALYSIS**

**2.1 Project Description**

The ONLINE VOTING SYSTEM-INDIA shall reduce the time spend making

long queues at the polling stations during voting. It shall also enable the

voters to vote from any part of the globe as explained since this is an

online application available on the internet. Cases of vote miscounts shall

also be solved since at the backend of this system resides a well developed

database using MYSQL that can provide the correct data once it’s correctly

queried. Since the voting process shall be open as early as possible, the

voters shall have ample time to decide when and whom to vote for.

**2.2 Requirement Specifications**

1) Registration of the voter is done by ELECTION COMMISION OF INDIA.

2) ELECTION COMMISION OF INDIA can change the information any time

if required.

3) Registration of the Voter depends upon the information filled by the

user.

4) Voter is given a unique ID and PASSWORD.

5) In the DATABASE information of every voter is stored.

6) Database shows the information of every user.

**Problems with the Existing Voter Registration System**

The problems of the existing manual system of voting include among others the following:

**2.2.1.** **Expensive and Time consuming**: The process of collecting data and

entering this data into the database takes too much time and is expensive

to conduct, for example, time and money is spent in printing data capture

forms, in preparing registration stations together with human resources,

and there after advertising the days set for registration process including

sensitizing voters on the need for registration, as well as time spent on

entering this data to the database.

**2.2.2** **Too much paper work:** The process involves too much paper work and

paper storage which is difficult as papers become bulky with the

population size.

**2.2.3. Errors during data entry**:Errors are part of all human beings; it is very unlikely for humans to be 100 percent efficient in data entry.

**2.2.4.** **Loss of registration forms**:Some times, registration forms get lost after

being filled in with voters’ details, in most cases these are difficult to

follow-up and therefore many remain unregistered even though they are

voting age nationals and interested in exercising their right to vote.

**2.2.5.** **Short time provided to view the voter register**:This is a very big problem

since not all people have free time during the given short period of time to

check and update the voter register.

**2.2.6**. Above all, a number of voters end up being locked out from voting.

**2.3 System Constraints**

**2.3.1 Software Requirements**

**Development:**

**.Software:-**Jetbrains Pycharm Community Edition 2019 2.2 x64

**.Operating System:-**Windows 10 pro

**.Processor:-**2.5 GHz

**.RAM:-**4 GB

**.ROM:-**1 TB

**Deployment:**

**. Software:-**web browser

**.Operating System:-**windows 2003 and above versions

**.Processor:-**133 MHz and above

**.RAM:-**2 GB and above

**.ROM:-** 1 TB recommened

**2.3.2 Hardware requirements**

Hard Disk Drive:32 GB

System Type :- 64-bit operating system

Peripherals:

Monitor

Mouse

Keyboard

**3.PROJECT DESIGN**

**3.1 Introduction to UML :-**

Unified Modelling Language (UML) is a general purpose modelling language. The main aim of UML is to define a standard way to visualize the way a system has been designed. It is quite similar to blueprints used in other fields of engineering.

UML is not a programming language, it is rather a visual language. We use UML diagrams to portray the behaviour and structure of a system. UML helps software engineers, businessmen and system architects with modelling, design and analysis. The Object Management Group (OMG) adopted Unified Modelling Language as a standard in 1997. Its been managed by OMG ever since. International Organization for Standardization (ISO) published UML as an approved standard in 2005. UML has been revised over the years and is reviewed periodically.

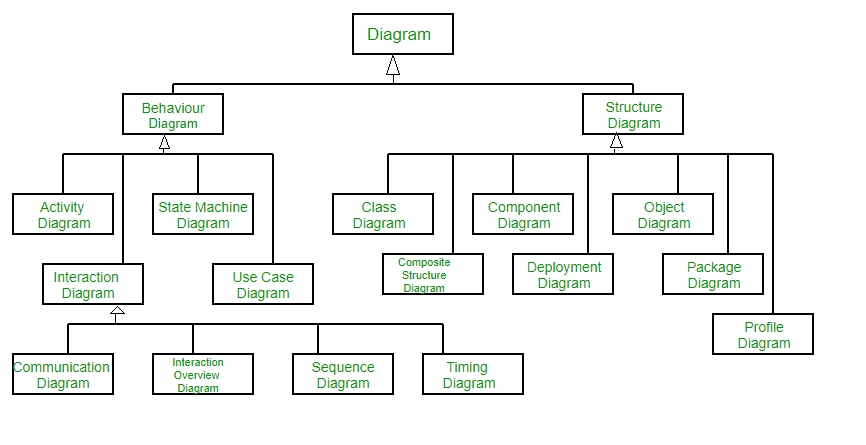
UML is linked with object oriented design and analysis. UML makes the use of elements and forms associations between them to form diagrams. Diagrams in UML can be broadly classified as:

1. **Structural Diagrams –** Capture static aspects or structure of a system. Structural Diagrams include: Component Diagrams, Object Diagrams, Class Diagrams and Deployment Diagrams.
2. **Behaviour Diagrams –** Capture dynamic aspects or behaviour of the system. Behaviour diagrams include: Use Case Diagrams, State Diagrams, Activity Diagrams and Interaction Diagrams.

**Object Oriented Concepts Used in UML –**

1. **Class –** A class defines the blue print i.e. structure and functions of an object.
2. **Objects –** Objects help us to decompose large systems and help us to modularize our system. Modularity helps to divide our system into understandable components so that we can build our system piece by piece. An object is the fundamental unit (building block) of a system which is used to depict an entity.
3. **Inheritance –** Inheritance is a mechanism by which child classes inherit the properties of their parent classes.
4. **Abstraction –** Mechanism by which implementation details are hidden from user.
5. **Encapsulation –** Binding data together and protecting it from the outer world is referred to as encapsulation.
6. **Polymorphism –** Mechanism by which functions or entities are able to exist in different forms.

The image below shows the hierarchy of diagrams according to UML 2.2



**Figure 3.1 (UML hierarchy)**

**Structural UML Diagrams –**

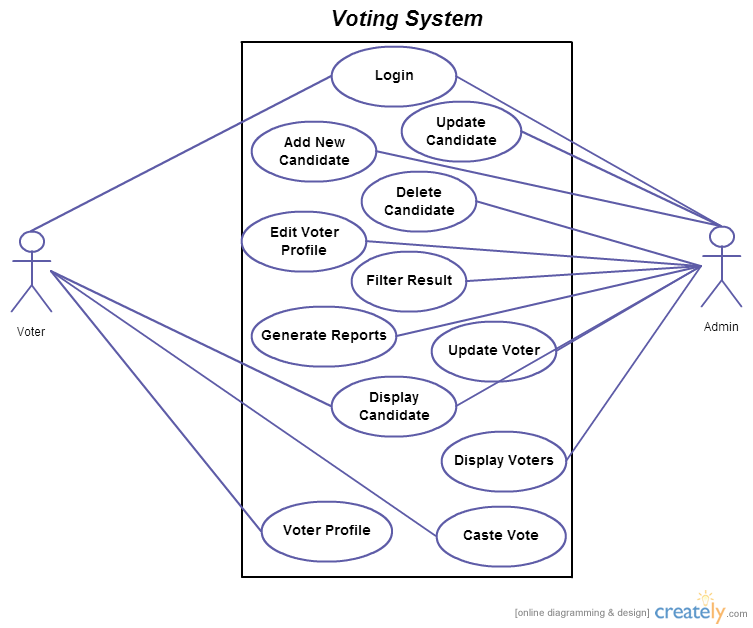
1. **Class Diagram –** The most widely use UML diagram is the class diagram. It is the building block of all object oriented software systems. We use class diagrams to depict the static structure of a system by showing system’s classes, their methods and attributes. Class diagrams also help us identify relationship between different classes or objects.
2. **Composite Structure Diagram –** We use composite structure diagrams to represent the internal structure of a class and its interaction points with other parts of the system. A composite structure diagram represents relationship between parts and their configuration which determine how the classifier (class, a component, or a deployment node) behaves. They represent internal structure of a structured classifier making the use of parts, ports, and connectors. We can also model collaborations using composite structure diagrams. They are similar to class diagrams except they represent individual parts in detail as compared to the entire class.
3. **Object Diagram –** An Object Diagram can be referred to as a screenshot of the instances in a system and the relationship that exists between them. Since object diagrams depict behaviour when objects have been instantiated, we are able to study the behaviour of the system at a particular instant. An object diagram is similar to a class diagram except it shows the instances of classes in the system. We depict actual classifiers and their relationships making the use of class diagrams. On the other hand, an Object Diagram represents specific instances of classes and relationships between them at a point of time.
4. **Component Diagram –** Component diagrams are used to represent the how the physical components in a system have been organized. We use them for modelling implementation details. Component Diagrams depict the structural relationship between software system elements and help us in understanding if functional requirements have been covered by planned development. Component Diagrams become essential to use when we design and build complex systems. Interfaces are used by components of the system to communicate with each other.
5. **Deployment Diagram –** Deployment Diagrams are used to represent system hardware and its software. It tells us what hardware components exist and what software components run on them. We illustrate system architecture as distribution of software artifacts over distributed targets. An artifactis the information that is generated by system software. They are primarily used when a software is being used, distributed or deployed over multiple machines with different configurations.
6. **Package Diagram –** We use Package Diagrams to depict how packages and their elements have been organized. A package diagram simply shows us the dependencies between different packages and internal composition of packages. Packages help us to organise UML diagrams into meaningful groups and make the diagram easy to understand. They are primarily used to organise class and use case diagrams.

**Behaviour Diagrams –**

1. **State Machine Diagrams –** A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It’s a behavioural diagram and it represents the behaviour using finite state transitions. State diagrams are also referred to as State machines and State-chart Diagrams. These terms are often used interchangeably. So simply, a state diagram is used to model the dynamic behaviour of a class in response to time and changing external stimuli.
2. **Activity Diagrams –** We use Activity Diagrams to illustrate the flow of control in a system. We can also use an activity diagram to refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on condition of flow and the sequence in which it happens. We describe or depict what causes a particular event using an activity diagram.
3. **Use Case Diagrams –** Use Case Diagrams are used to depict the functionality of a system or a part of a system. They are widely used to illustrate the functional requirements of the system and its interaction with external agents(actors). A use case is basically a diagram representing different scenarios where the system can be used. A use case diagram gives us a high level view of what the system or a part of the system does without going into implementation details.
4. **Sequence Diagram –** A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.
5. **Communication Diagram –** A Communication Diagram(known as Collaboration Diagram in UML 1.x) is used to show sequenced messages exchanged between objects. A communication diagram focuses primarily on objects and their relationships. We can represent similar information using Sequence diagrams, however, communication diagrams represent objects and links in a free form.
6. **Timing Diagram –** Timing Diagram are a special form of Sequence diagrams which are used to depict the behaviour of objects over a time frame. We use them to show time and duration constraints which govern changes in states and behaviour of objects.

**4.PROJECT DIAGRAM**

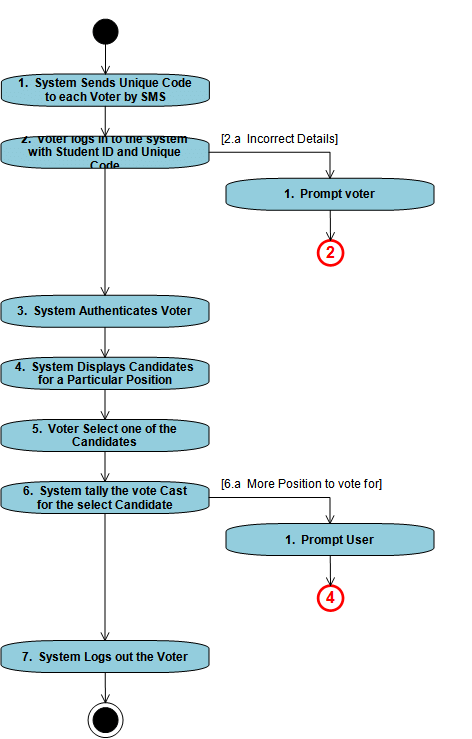
**4.1 USE-CASE DAGRAM**

****

Use Case Diagrams are used to depict the functionality of a system or a part of a system. They are widely used to illustrate the functional requirements of the system and its interaction with external agents(actors). A use case is basically a diagram representing

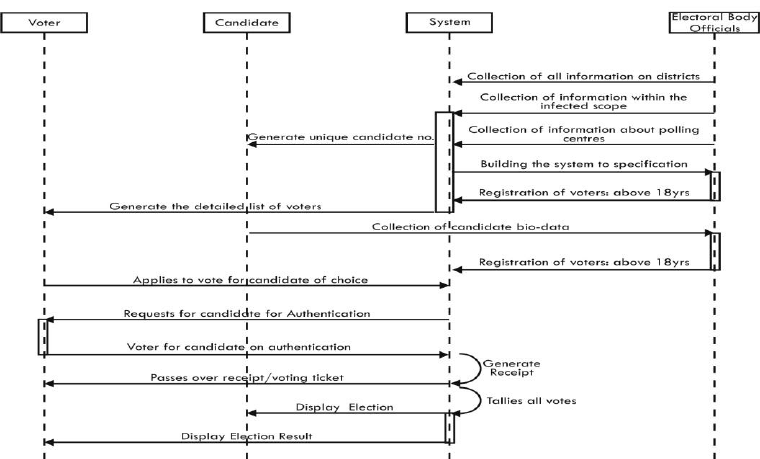
different scenarios where the system can be used. A use case diagram gives us a high level view of what the system or a part of the system does without going into implementation details.

**4.2 ACTIVITY DIAGRAM**

****

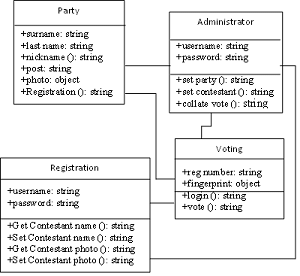
An **activity diagram** visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. Activity diagrams are often used in business process modeling. They can also describe the steps in a use case diagram. Activities modeled can be sequential and concurrent.

**4.3 SEQUENCE DIAGRAM**

****

A **sequence diagram** simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequencediagram. Sequence diagrams describe how and in what order the objects in a system function.

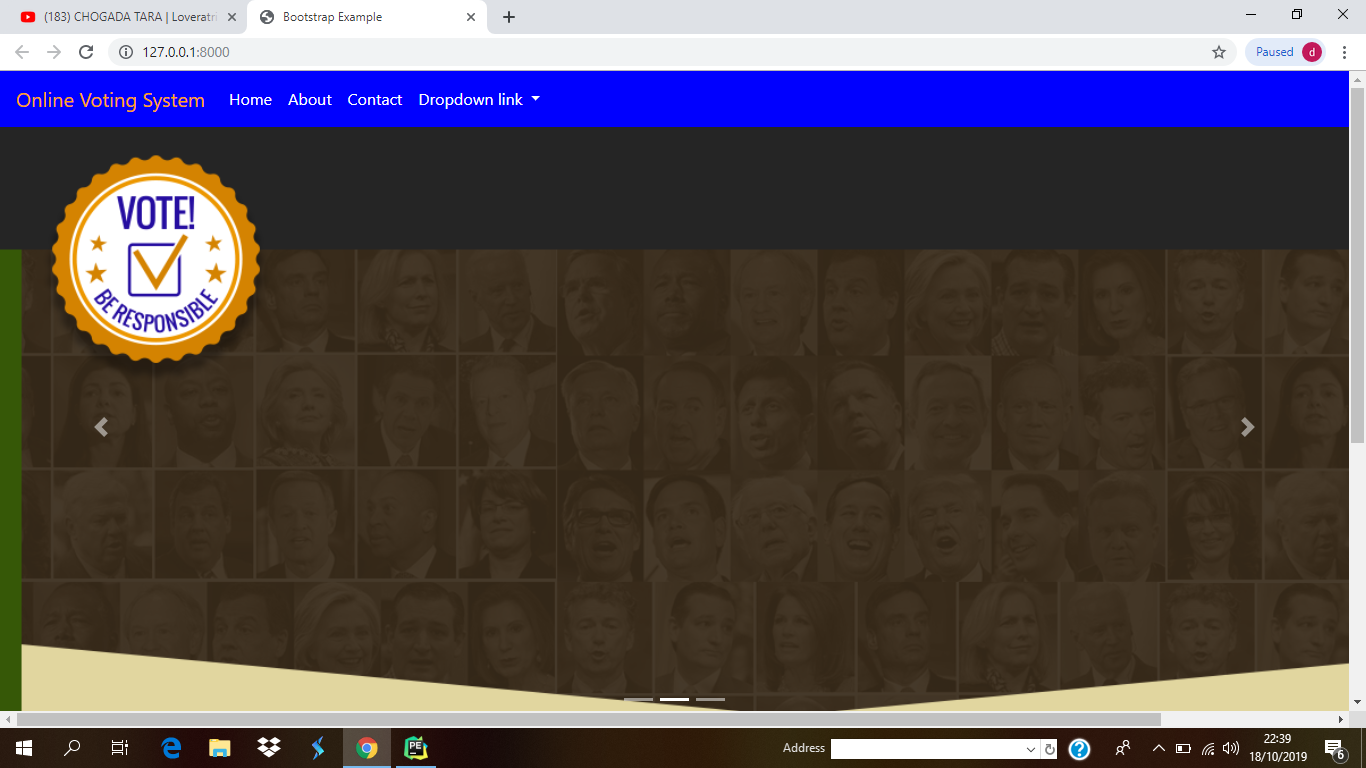
**4.4 CLASS DIAGRAM**

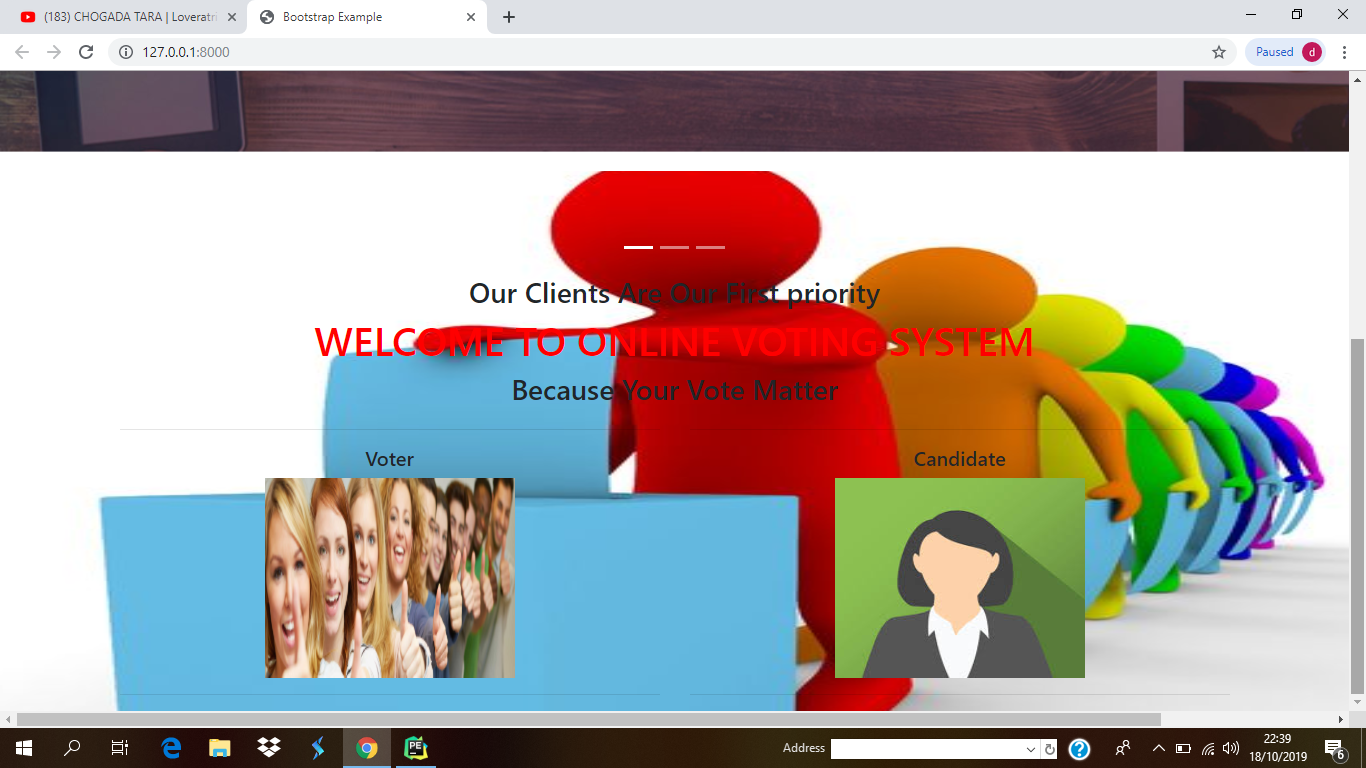
****

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

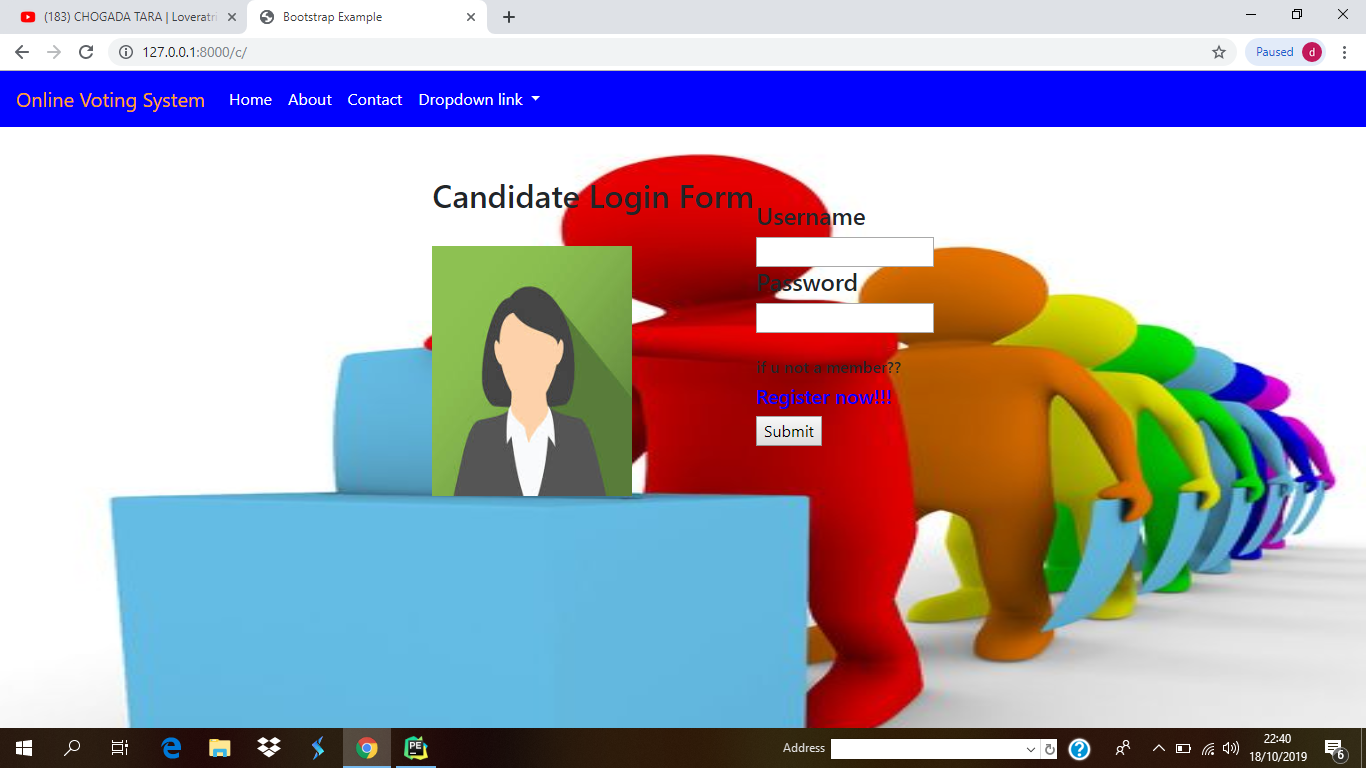
1. **SNAPSHOTS**

**5.1 Home Page**

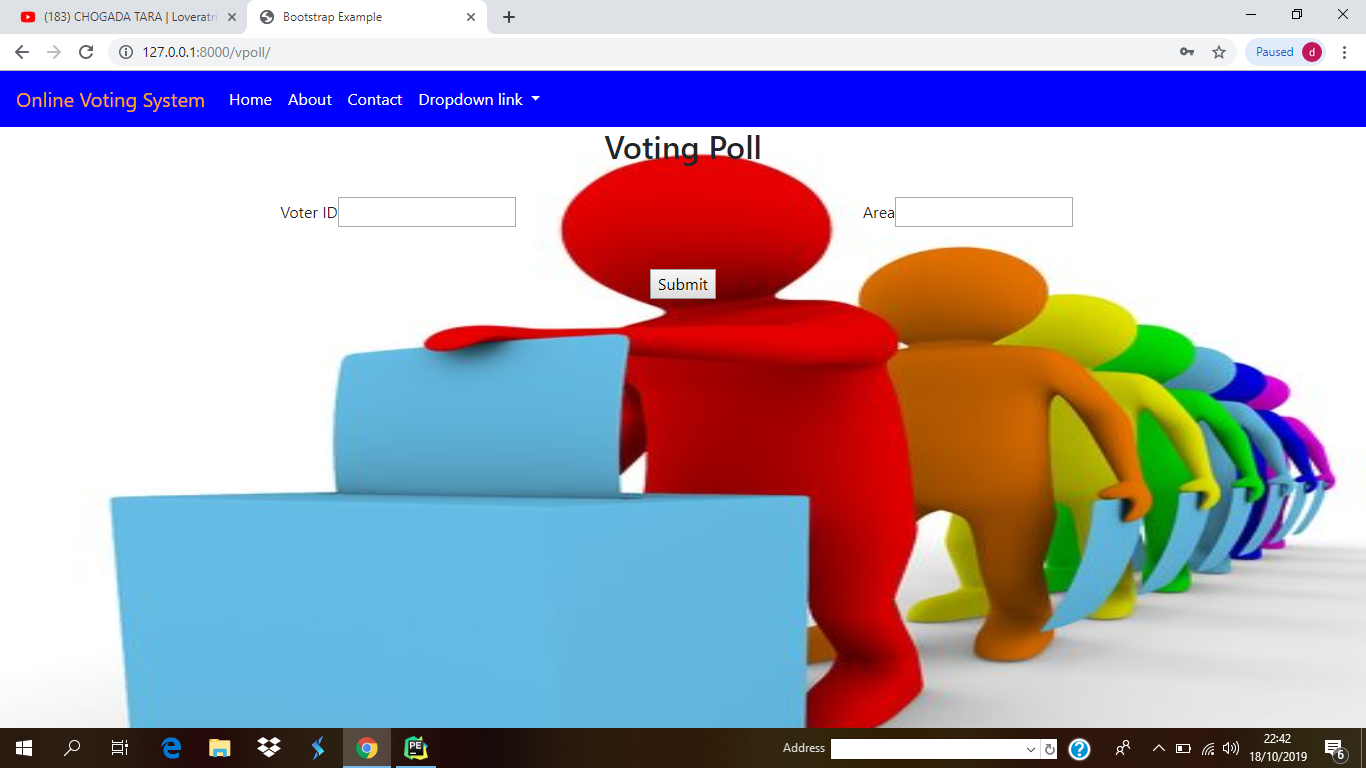
****

****

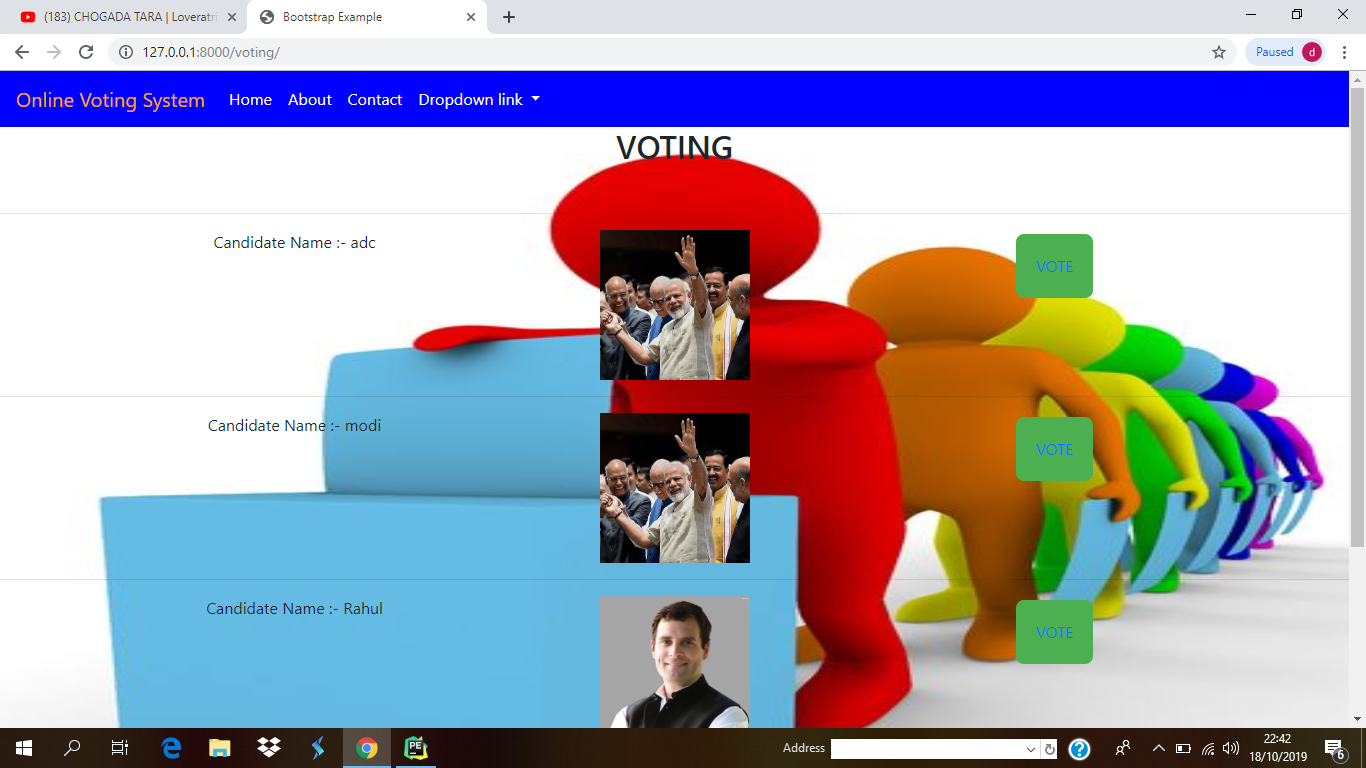
**Candidate Login Form**

****

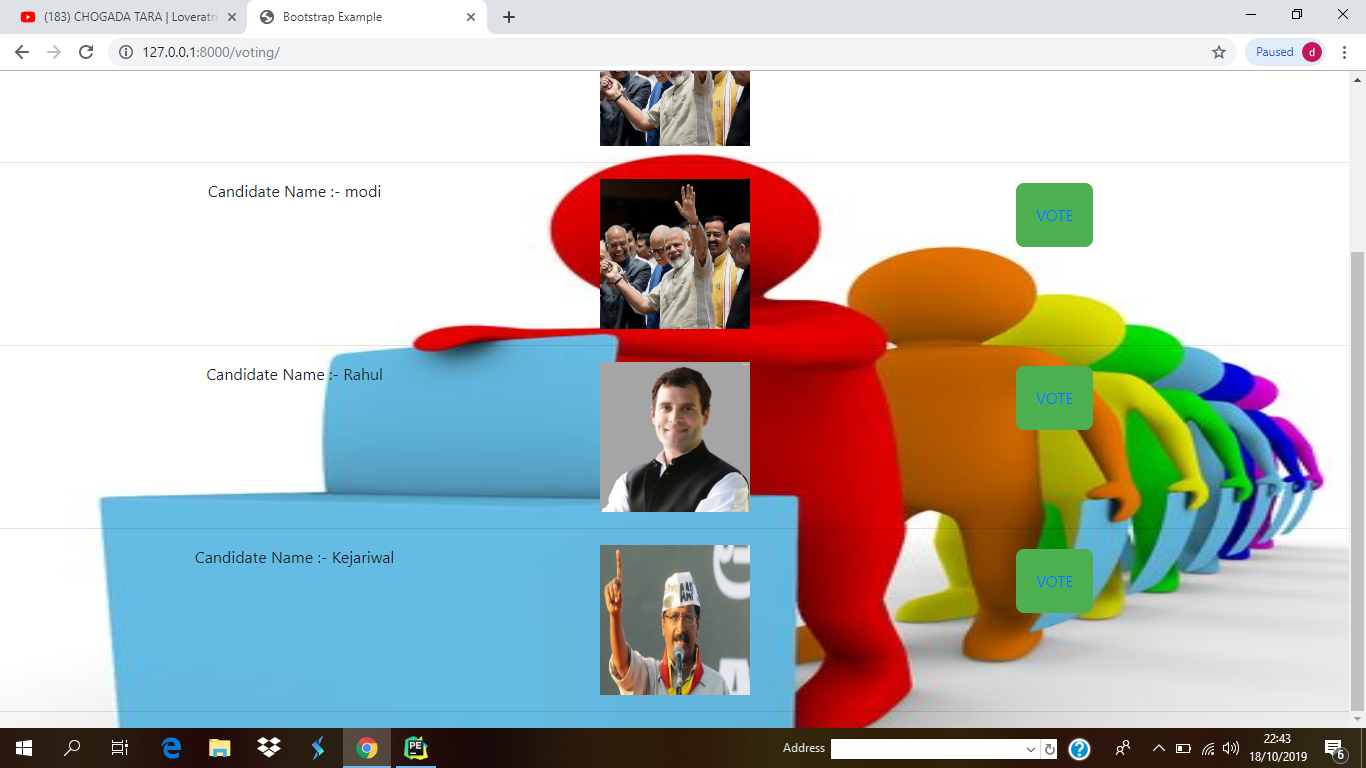
**Voting Poll**

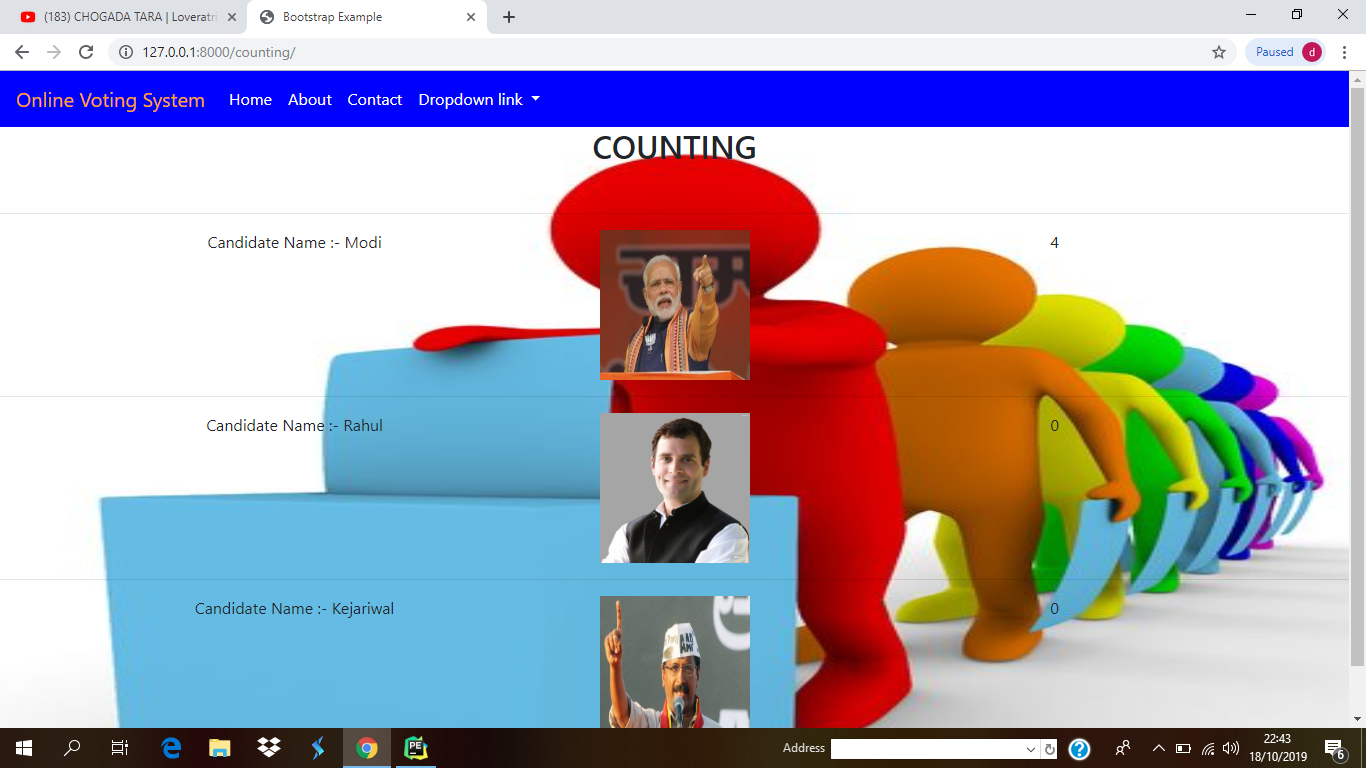
****

**Voting Day**

****

**Result Day**

****

****

**6 TESTING**

**6.1 Software Testing :-**

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

According to ANSI/IEEE 1059 standard, Testing can be defined as - A process of analysing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.

It depends on the process and the associated stakeholders of the project(s). In the IT industry, large companies have a team with responsibilities to evaluate the developed software in context of the given requirements. Moreover, developers also conduct testing which is called **Unit Testing**. In most cases, the following professionals are involved in testing a system within their respective capacities −

* Software Tester
* Software Developer
* Project Lead/Manager
* End User

Different companies have different designations for people who test the software on the basis of their experience and knowledge such as Software Tester, Software Quality Assurance Engineer, QA Analyst, etc.

It is not possible to test the software at any time during its cycle. The next two sections state when testing should be started and when to end it during the SDLC.

## When to Start Testing?

An early start to testing reduces the cost and time to rework and produce error-free software that is delivered to the client. However in Software Development Life Cycle (SDLC), testing can be started from the Requirements Gathering phase and continued till the deployment of the software.

It also depends on the development model that is being used. For example, in the Waterfall model, formal testing is conducted in the testing phase; but in the incremental model, testing is performed at the end of every increment/iteration and the whole application is tested at the end.

Testing is done in different forms at every phase of SDLC −

* During the requirement gathering phase, the analysis and verification of requirements are also considered as testing.
* Reviewing the design in the design phase with the intent to improve the design is also considered as testing.
* Testing performed by a developer on completion of the code is also categorized as testing.

## When to Stop Testing?

It is difficult to determine when to stop testing, as testing is a never-ending process and no one can claim that a software is 100% tested. The following aspects are to be considered for stopping the testing process −

* Testing Deadlines
* Completion of test case execution
* Completion of functional and code coverage to a certain point
* Bug rate falls below a certain level and no high-priority bugs are identified
* Management decision

## Verification & Validation

These two terms are very confusing for most people, who use them interchangeably. The following table highlights the differences between verification and validation.

|  |  |  |
| --- | --- | --- |
| **Sr.No.** | **Verification** | **Validation** |
| 1 | Verification addresses the concern: "Are you building it right?" | Validation addresses the concern: "Are you building the right thing?" |
| 2 | Ensures that the software system meets all the functionality. | Ensures that the functionalities meet the intended behaviour. |
| 3 | Verification takes place first and includes the checking for documentation, code, etc. | Validation occurs after verification and mainly involves the checking of the overall product. |
| 4 | Done by developers. | Done by testers. |
| 5 | It has static activities, as it includes collecting reviews, walkthroughs, and inspections to verify a software. | It has dynamic activities, as it includes executing the software against the requirements. |
| 6 | It is an objective process and no subjective decision should be needed to verify a software. | It is a subjective process and involves subjective decisions on how well a software works. |

There are different methods that can be used for software testing. This chapter briefly describes the methods available.

## Black-Box Testing

The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

The following table lists the advantages and disadvantages of black-box testing.

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Well suited and efficient for large code segments. | Limited coverage, since only a selected number of test scenarios is actually performed. |
| Code access is not required. | Inefficient testing, due to the fact that the tester only has limited knowledge about an application. |
| Clearly separates user's perspective from the developer's perspective through visibly defined roles. | Blind coverage, since the tester cannot target specific code segments or errorprone areas. |
| Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language, or operating systems. | The test cases are difficult to design. |

## White-Box Testing

White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called **glass testing** or **open-box testing**. In order to perform **white-box** testing on an application, a tester needs to know the internal workings of the code.

The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

The following table lists the advantages and disadvantages of white-box testing.

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| As the tester has knowledge of the source code, it becomes very easy to find out which type of data can help in testing the application effectively. | Due to the fact that a skilled tester is needed to perform white-box testing, the costs are increased. |
| It helps in optimizing the code. | Sometimes it is impossible to look into every nook and corner to find out hidden errors that may create problems, as many paths will go untested. |
| Extra lines of code can be removed which can bring in hidden defects. | It is difficult to maintain white-box testing, as it requires specialized tools like code analysers and debugging tools. |
| Due to the tester's knowledge about the code, maximum coverage is attained during test scenario writing. |  |

## Grey-Box Testing

Grey-box testing is a technique to test the application with having a limited knowledge of the internal workings of an application. In software testing, the phrase the more you know, the better carries a lot of weight while testing an application.

Mastering the domain of a system always gives the tester an edge over someone with limited domain knowledge. Unlike black-box testing, where the tester only tests the application's user interface; in grey-box testing, the tester has access to design documents and the database. Having this knowledge, a tester can prepare better test data and test scenarios while making a test plan.

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Offers combined benefits of black-box and white-box testing wherever possible. | Since the access to source code is not available, the ability to go over the code and test coverage is limited. |
| Grey box testers don't rely on the source code; instead they rely on interface definition and functional specifications. | The tests can be redundant if the software designer has already run a test case. |
| Based on the limited information available, a grey-box tester can design excellent test scenarios especially around communication protocols and data type handling. | Testing every possible input stream is unrealistic because it would take an unreasonable amount of time; therefore, many program paths will go untested. |
| The test is done from the point of view of the user and not the designer. |  |

**7. CONCLUSION**

This Online Voting system will manage the Voter’s information by which

voter can login and use his voting rights. The system will incorporate all features of Voting system. It provides the tools for maintaining voter’s vote to every party and it count total no. of votes of every party. There is a DATABASE which is maintained by the ELECTION COMMISION OF INDIA in which all the names of voter with complete information is stored.

In this user who is above 18 year’s register his/her information on the

database and when he/she want to vote he/she has to login by his id and

password and can vote to any party only single time. Voting detail store in

database and the result is displayed by calculation. By online voting system

percentage of voting is increases. It decreases the cost and time of voting

process. It is very easy to use and It is vary less time consuming. It is very easy to debug.

**REFERENCES**

**Websites**

* <https://www.tutorialspoint.com/software_testing/software_testing_methods.htm>
* <https://www.tutorialspoint.com/uml/uml_standard_diagrams.htm>
* <https://www.w3schools.com>
* <https://www.tutorialspoint.com/django/index.htm>
* <https://www.javatpoint.com/django-tutorial>