ABSTRACT

The project's main objective is to enabling voting at elsewhere regardless of an individual's current location through online mode providing a secured and user-friendly Voting System. The problem of e-voting is still critical in terms of safety and security. This is a robust system, which deals with the design and development of a web-based voting system. Our system deals with online voting system that facilitates user(voter), candidate and administrator (who will be in charge to monitor voting) to participate in online voting. our online voting system is highly secured, and it has a simple and interactive user interface.

The proposed online portal is secured and have unique security feature such as unique id generation that adds another layer of security (except login id and password) and the voter's data are predefined in the database, which helps in authentication. It also creates and manages voting and an election detail as all the users must login by user name and password (as their date of birth) and click on candidates to register vote.

The voters and nominees are verified through OTP each and every time of their login. This system eliminates any intruders or duplicate logins.

The project Online Voting Software aims at making the voting process easy in any type of elections. Presently voting is performed using ballot paper and the counting is done manually, hence it consumes a lot of time. There can be possibility of invalid votes. All these make election a tedious task. In our proposed system voting and counting is done with the help of computer in Online. It saves time, avoid error in counting and there will be no invalid votes. It also avoids the process of physical touching or visiting any places and so in the time of pandemic too it will be more helpful to conduct elections. Without the wastage of time the citizen can vote the respective candidate. So, there is a need for Online Voting Systems.

CHAPTER 1

INTRODUCTION

This chapter starts with a high-level overview of the project. It then describes the specific aims and objectives of the project. Finally, it analyses the feasibility of the project and provides a feasibility report of the system.

1.1 Background of Study

In every democratic setting with persons of differing and inconsistent opinions, decisions must be made between several options. This happens in business environment, educational environment, social organizations, and mostly in governance. One of the ways of making such a decision is through voting. Voting is a formal process of expressing individual opinions for or against some motion. In the governance sector of many organizations this process is always used as a means of selecting or electing a leader. One of the key areas where voting is applied is in election. Election is the formal process of selecting a person for public office or of accepting or rejecting a political proposition by voting.

1.1.1E-voting System Overview

The proposed online portal is secured and have unique security feature such as unique id generation that adds another layer of security (except login id and password) and the voter’s data are predefined in the database, which helps in authentication. It also creates and manages voting and an election detail as all the users must login by user name and password (as their date of birth) and click on candidates to register vote. The voters and nominees are verified through OTP each and every time of their login. This system eliminates any intruders or duplicate logins. In our proposed system voting and counting is done with the help of computer in Online. It saves time, avoid error in counting and there will be no invalid votes. It also avoids the process of physical touching or visiting any places and so in the time of pandemic too it will be more helpful to conduct elections. Without the wastage of time the citizen can vote the respective candidate. So, there is a need for Online Voting Systems.

1.2 Problem Statement

The present voting system applicable in the India electoral system has proved inefficient as the voters' registration process is slow, the manual collation of results takes time and gives room for result manipulation, also the inaccessible nature of election venues which includes the long distances to be covered by voters' to their registered location increases voters' apathy towards the election processes, and finally the issues of ballot box snatching and damage and other election violence and issues associated with the traditional ballot paper voting all defiles the purpose of voting in election process as a formal process of expressing individual opinions for or against some motion.

1.3 Aim and Objectives

In the quest to design a successful system to tackle the issues stated in the problem statement, the aim and objectives of the project are outlined below.

1.3.1Aim

The aim of this project is to design and implement a user-friendly real-time e-voting system.

1.3.2 Objectives

Project Objectives includes

1. A detailed study of the election processes as it pertains to voting.
2. Design and develop a software platform for voter registration, election voting, real-time election results collation and monitoring and mostly for voters' remote access to elections.
3. Design and develop an software system that incorporates OTP and personal information Authentication to the voters via e-mail.
4. Design and develop an administration dashboard for the election administrators.
5. Run simulations and compare the results of the designed e-voting system and other voting systems.

1.4 Significance of the Project

In view of the rapid development of computer technology in virtually all fields of operation and its use in relation to information management, the projects' benefits are itemized as follows:

1.4.1To the University

An e-voting system is beneficial to the university as:

1. It will provide a means conduct a less stressful and fair elections at different levels (faculty, departments, school wide etc.) in the university.
2. It will offer an in-depth knowledge of the practical approach to ICT education.
3. It will serve as a hands-on application of theories taught in class as it relates to database, software and hardware development.
4. As its' database is based on a flexible database management system, student and staff details can easily be collected for easy access and monitoring.
5. It will serve as a base for other works in the field of ICT in governance.

1.4.2To the Society

The significance of an e-voting system to the society and mostly to Nigeria are itemized as follows:

1. It will provide ECI (Election commission of India) with a means to conduct less costly and fair elections.
2. The secure and flexible database management system safeguards data and information to account for credible elections.
3. It will serve to reduce the workload in the process of conducting election.
4. As it incorporates online voting individuals can vote from their convenience.
5. It will enable ECI reduce the time wasted in collating and announcing election result.
6. It will greatly reduce and eliminate disenfranchising electorates.
7. It will serve to eliminate invalid votes, curb election violence as votes are counted immediately as they are cast.

1.5 Scope/Limitations of the Work

This project work is mainly designed to enable the Independent National Electoral Commission to use electronic device to capture voter's information, and to allow voters to their cast votes easily and comfortably to promote a more credible election which is efficient and less costly. The dynamic nature of the elections application interface and database structure allows for different organizations set up and conduct basic elections too. Its online interface enables real-time election monitoring and result collation. Some of its major limitations are:

- ❖ It requires network access: Since the collection and sending of votes to the database requires an internet access which may not be readily available in some urban area would seem a limiting factor, though the local database and the printed vote can be used for counting until network is restored.

1.6 Project Outline

This project work on e-voting system is made up of five chapters: introduction, literature review, methodology and system design, systems implementation and result analysis, conclusion and recommendation.

In the chapter one of this project, the introduction which briefly explains voting and elections in general, is seen. It goes further to explain the background of an e-voting system, the aim and objectives of the e-voting system, its significance, scope, and constraints. It summaries by giving the project outline.

In the chapter two, a review of previous literature and technologies used for e-voting system was treated. We also see the different approaches to e-voting systems, their implementation, criticism with their literature reviews and noted the various gaps in the existing literatures.

In chapter three, we see the block diagram of the project work, different methodologies used in development stages, the different phases of the project work which include its research, design, display programming, testing. We extensively cover the requirements of the project designs and software incorporated in the work.

In chapter four, we talk about the steps taken and techniques used for the actual implementation of the project. We see tests carried out to ensure that the project is efficient and also display the result gotten and their significance. We also see the problems encountered and the techniques and solutions taken to overcome them or not.

And finally, we conclude the work and give notable recommendations for optimal operation of the product. Also, we provide suggestions for improvement, enhancement and optimization of our existing work. We also outline the major contribution to the body of knowledge in which our work has achieved

CHAPTER 2

LITERATURE REVIEW

This chapter discusses the different approaches to e-voting processes. After that, it explains the web application development tools and technologies used in this project.

2.1 Web Application Based System

A web-based system is an application that is accessed via HTTP. The term web-based is usually used to describe applications that run in a web browser. It can, though, also be used to describe applications that have a very small component of the solution loaded on the client's PC. The host server for a web-based system could be a local server, or it could be accessed via the internet.
Web-based applications used to be very limited in functionality. However, advances in technology, security, and internet speeds have greatly increased the potential scope of web-based systems. Today, we have web-based business accounting systems, web-based CRM systems, a web-based Microsoft Office, and more. Web-based applications offer some significant advantages over native, client-based software. Here are just some of the benefits of web-based apps for business.

2.2 Web Application Development Technologies

This section is providing a brief description of the technologies used for this project.

2.2.1 Docker

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

2.2.1.1 Docker Platform

Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allow you to run many containers simultaneously on a given host. Containers are lightweight and contain everything needed to run the application, so you do not need to rely on what is currently installed on the host. You can easily share containers while you work, and be sure that everyone you share with gets the same container that works in the same way.

Docker provides tooling and a platform to manage the lifecycle of your containers:

- Develop your application and its supporting components using containers.
- The container becomes the unit for distributing and testing your application.
- When you're ready, deploy your application into your production environment, as a container or an orchestrated service. This works the same whether your production environment is a local data centre, a cloud provider, or a hybrid of the two.

2.2.1.2 Docker Architecture

Docker uses a client-server architecture. The Docker *client* talks to the Docker *daemon*, which does the heavy lifting of building, running, and distributing your Docker containers. The Docker client and daemon *can* run on the same system, or you can connect a Docker client to a remote Docker daemon. The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface. Another Docker client is Docker Compose, that lets you work with applications consisting of a set of containers.

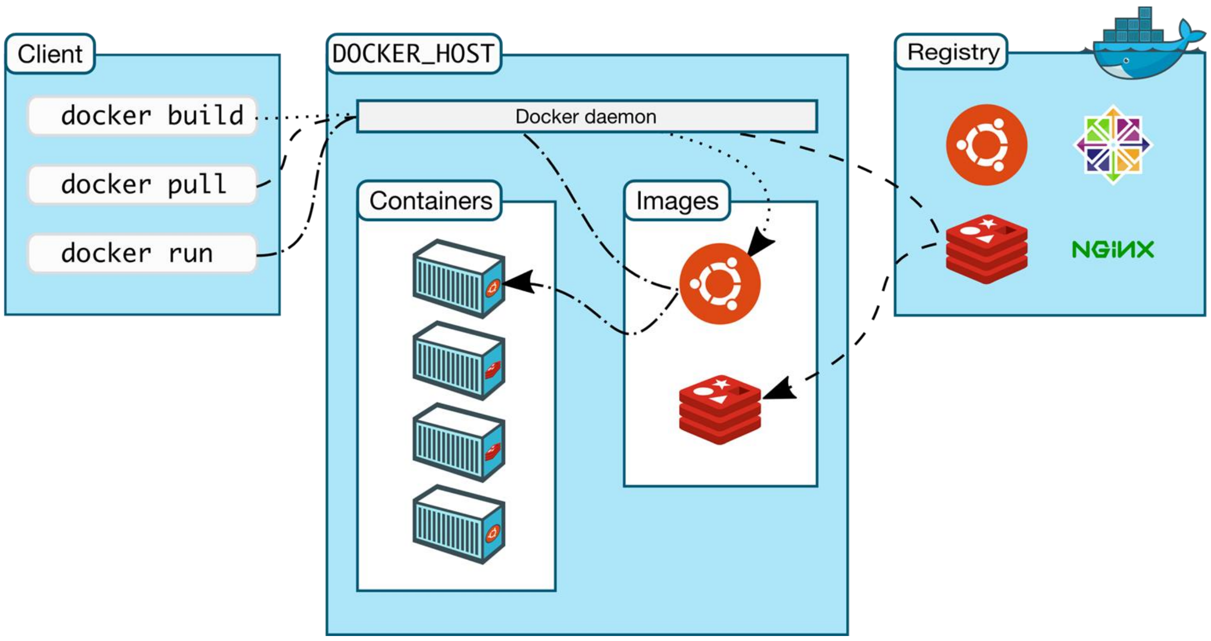

Figure 1: *Docker architecture*

2.2.2 MERN Stack

MERN Stack is a JavaScript Stack that is used for easier and faster deployment of full-stack web application. MERN Stack comprises of 4 technologies namely: MongoDB, Express.js, React.js and Node.js. It is designed make the development process smoother and easier.

Each of these 4 powerful technologies provides an end-to-end framework for the developer to work in and each of these technologies play a big part in the development of web applications.

- **MongoDB** - Document Database.
- **Express(.js)** - Node.js web framework.
- **React(.js)** - client-side JavaScript framework.
- **Node(.js)** - The premier JavaScript web server

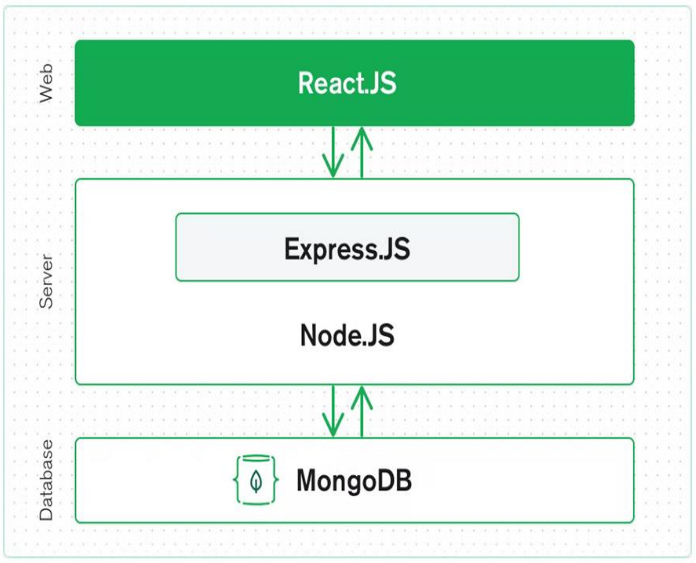

Figure 2: Illustrative diagram of the MERN stack

2.2.2.1 MongoDB

MongoDB is a NoSQL database where each record is a document comprising of key-value pairs that are similar to JSON (JavaScript Object Notation) objects. MongoDB is flexible and allows its users to create schema, databases, tables, etc. Documents that are identifiable by a primary key make up the basic unit of MongoDB. Once MongoDB is installed, users can make use of Mongo shell as well. Mongo shell provides a JavaScript interface through which the users can interact and carry out operations (e.g.: querying, updating records, deleting records).

MongoDB Atlas Cluster is a NoSQL Database-as-a-Service offering in the public cloud (available in Microsoft Azure, Google Cloud Platform, Amazon Web Services). This is a managed MongoDB service, and with just a few clicks, you can set up a working MongoDB cluster, accessible from your favourite web browser.

You don't need to install any software on your workstation as you can connect to MongoDB directly from the web user interface as well as inspect, query, and visualize data.

Why use MongoDB?

- ❖ Fast – Being a document-oriented database, easy to index documents. Therefore, a faster response.
- ❖ Scalability – Large data can be handled by dividing it into several machines.
- ❖ Use of JavaScript – MongoDB uses JavaScript which is the biggest advantage.
- ❖ Schema Less – Any type of data in a separate document.
- ❖ Data stored in the form of JSON –
    1. Objects, Object Members, Arrays, Values, and Strings
    2. JSON syntax is very easy to use.
    3. JSON has a wide range of browser compatibility.
    4. Sharing Data: Data of any size and type (video, audio) can be shared easily.
- ❖ Simple Environment Setup – It's really simple to set up MongoDB.
- ❖ Flexible Document Model – MongoDB supports document-model (tables, schemas, columns & SQL) which is faster and easier.

2.2.2.2 Express.js

Express is a Node.js framework. Rather than writing the code using Node.js and creating loads of Node modules, Express makes it simpler and easier to write the back-end code. Express helps in designing great web applications and APIs. Express supports many middleware's which makes the code shorter and easier to write.

Why use Express?

- ❖ Asynchronous and Single-threaded.
- ❖ Efficient, fast & scalable
- ❖ Has the biggest community for Node.js
- ❖ Express promotes code reusability with its built-in router.
- ❖ Robust API
- ❖ Create a new folder to start your express project and type below command in the command prompt to initialize a package.json file. Accept the default settings and continue.

2.2.2.3 React.js

React is a JavaScript library that is used for building user interfaces. React is used for the development of single-page applications and mobile applications because of its ability to handle rapidly changing data. React allows users to code in JavaScript and create UI components.

Why use React?

- ❖ Virtual DOM – A virtual DOM object is a representation of a DOM object. Virtual DOM is actually a copy of the original DOM. Any modification in the web application causes the entire UI to re-render the virtual DOM. Then the difference between the original DOM and this virtual DOM is compared and the changes are made accordingly to the original DOM.
- ❖ JSX – Stands for JavaScript XML. It is an HTML/XML JavaScript Extension which is used in React. Makes it easier and simpler to write React components.

- Components – ReactJS supports Components. Components are the building blocks of UI wherein each component has a logic and contributes to the overall UI. These components also promote code reusability and make the overall web application easier to understand.
- High Performance – Features like Virtual DOM, JSX and Components makes it much faster than the rest of the frameworks out there.

2.2.2.4 Node.js

Node.js provides a JavaScript Environment which allows the user to run their code on the server (outside the browser). Node pack manager i.e., npm allows the user to choose from thousands of free packages (node modules) to download.

Why use Node.JS?

- Open-source JavaScript Runtime Environment
- Single threading – Follows a single-threaded model.
- Data Streaming
- Fast – Built on Google Chrome's JavaScript Engine, Node.js has a fast code execution.
- Highly Scalable
- Initialize a Node.js application by typing running the below command in the command window. Accept the standard settings.

2.2.3 CSS Framework

2.2.3.1 Tailwind CSS

Tailwind CSS can be used to make websites in the fastest and the easiest way. Tailwind CSS is basically a utility-first CSS framework for rapidly building custom user interfaces. It is a highly customizable, low-level CSS framework that gives you all of the building blocks you need to build bespoke designs without any annoying opinionated styles you have to fight to override.

The beauty of this thing called tailwind is it doesn’t impose design specification or how your site should look like, you simply bring tiny components together to construct a user interface that is unique. What Tailwind simply does is take a ‘raw’ CSS file, processes this CSS file over a configuration file, and produces an output.

Why Tailwind CSS

- Faster UI building process
- It is a utility-first CSS framework which means we can use utility classes to build custom designs without writing CSS as in traditional approach.

Advantages of Tailwind CSS

- No more silly names for CSS classes and Id’s.
- Minimum lines of Code in CSS file.
- We can customize the designs to make the components.
- Makes the website responsive.
- Makes the changes in the desired manner. CSS is global in nature and if make changes in the file the property is changed in all the HTML files linked to it. But with the help of Tailwind CSS, we can use utility classes and make local changes.

2.2.4 Front-end Library

2.2.4.1 Redux.js

Redux is an open-source JavaScript library for managing and centralizing application state. It is most commonly used with libraries such as React or Angular for building user interfaces. Similar to (and inspired by) Facebook's Flux architecture, it was created by Dan Abramov and Andrew Clark. Since mid-2016, the primary maintainers are Mark Erikson and Tim Dorr.

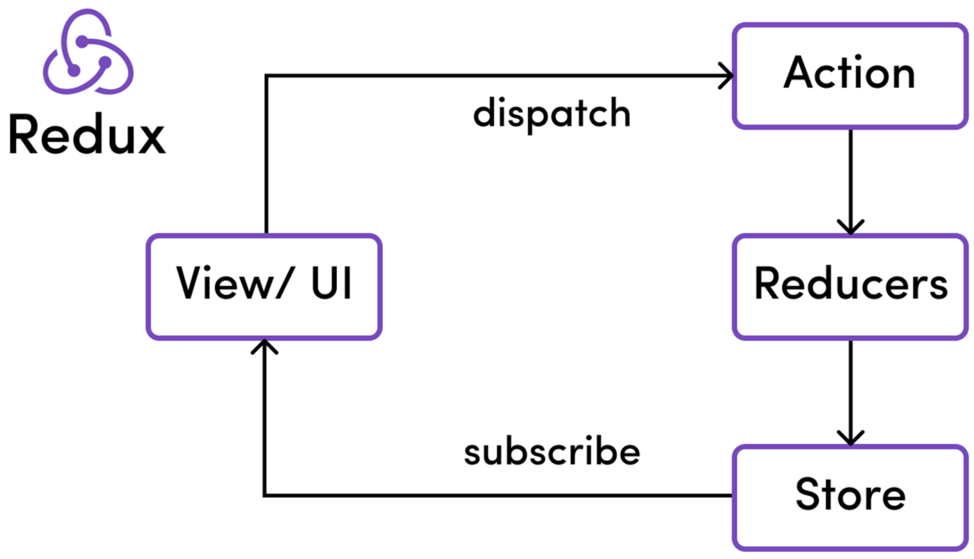

Figure 3: *Core principles of Redux*

Why use Redux?

When using Redux with React, states will no longer need to be lifted up. This makes it easier for you to trace which action causes any change.

As you can see in the example above, the component does not need to provide any state or method for its children components to share data among themselves. Everything is handled by Redux. This greatly simplifies the app and makes it easier to maintain.

2.2.5 Front-end Packages

Axios

Axios is a promised-based HTTP client for JavaScript. It has the ability to make HTTP requests from the browser and handle the transformation of request and response data.

Chart.js

Chart.js is a free, open-source JavaScript library for data visualization, which supports eight chart types: bar, line, area, pie (doughnut), bubble, radar, polar, and scatter. Chart.js renders in HTML5 canvas and is widely covered as one of the best data visualization libraries.

React-csv

Generate a CSV file from give data. This data can be an array of arrays, an array of literal objects, or strings.

React-icons

React-icons is a small library that helps you add icons (from all different icon libraries) to your React apps. It delivers the icons to your app as components so they’re easier to work with, and it lets you style them so they’re consistent with the overall style of your app.

2.2.6 Back-end Package

Bcrypt.js

The bcrypt hashing function allows us to build a password security platform that scales with computation power and always hashes every password with a salt.

Cloudinary

Clodinary is an end-to-end image and video management solution for website and mobile apps, covering everything from image and video uploads, storage, manipulations, optimizations to delivery.

CORS

Cross-Origin Resource Sharing (CORS) is an HTTP header-based mechanism that allows a server to indicate any origins (domain, scheme or port) other than its own from which a browser should permit loading resource. CORS also relies on mechanism by which browsers make a “pre-flight” request to the server hosting the cross-origin resource, in order to check that the server will permit the actual request. In the pre-flight, the browser sends headers that indicate the HTTP method and headers that will be used in the actual request.

EJS

EJS is a simple templating language that lets you generate HTML markup with plain JavaScript. No religiousness about how to organize things. No reinvention of iteration and control-flow. It’s just plain JavaScript.

JWT

JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained for security transmitting information between parties as a JSON object.

This information can be verified and trusted because it is digitally signed.

Multter

Multer is a node.js middleware for handling multipart/form-data, which is primarily used for uploading files. It is written on top of busboy for maximum efficiency.

Nodemailer

Nodemailer is a module for Node.js application to allow easy as cake email sending. The project got started back in 2010 when there was no sane option to send mail message, today it is the solution most Node.js users turn to by default. Npm install nodemailer.

CHAPTER THREE

METHODOLOGY

The methodology is the systematic, theoretical analysis of the methods applied to a field of study. The aim of this chapter is to give an introduction to the general waterfall methodology for development used in this project.

3.1 System Development Methodology

For developing any information system, a System Development Methodology should be used which will provide a structured way for the development of IT-based systems. SDLC refers to the System or Software Design Life Cycle. It is phases of processes taken down to build a system properly. The main aim of the SDLC process is to help provide a system that is effective, cost-efficient, and of high quality. SDLC methodologies typically have the following stages: Analysis (requirements and design), construction, testing, release, and maintenance (response). But the phases can be changed in different SDLC methodologies.

3.1.1 Waterfall Model

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The Waterfall model is the earliest SDLC approach that was used for software development. The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome

of one phase acts as the input for the next phase sequentially.
The following illustration is a representation of the different phases of the Waterfall Model.

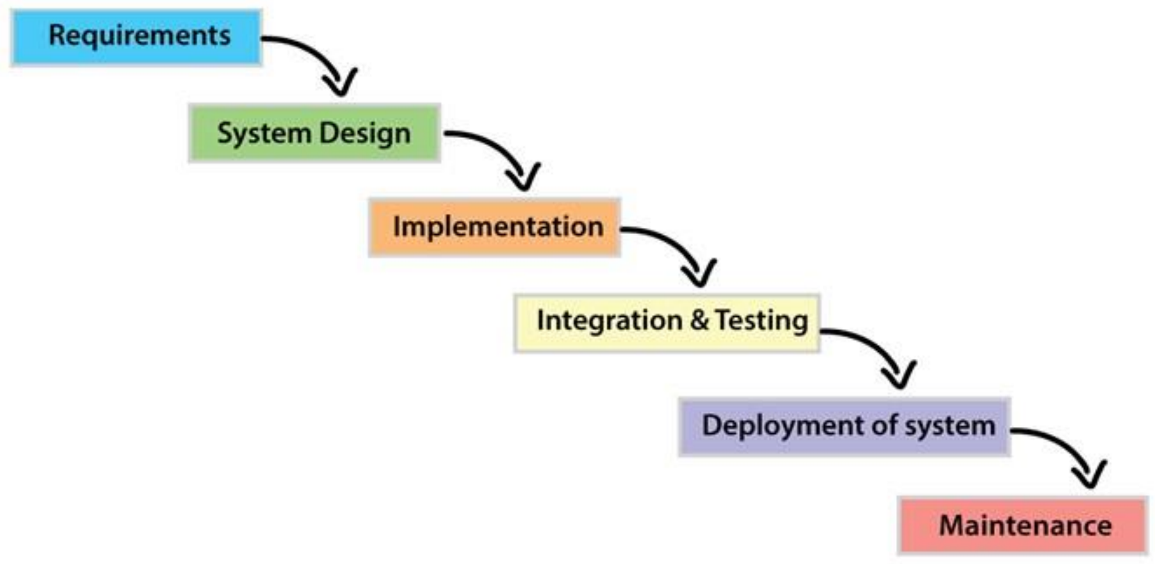

Figure 4: *Steps of a Waterfall model.*

Requirement Gathering and analysis:

All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

System Design:

The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying system requirements and helps in defining the overall system architecture.

Implementation:

With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

Integration and Testing:

All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

Deployment of system:

Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

Maintenance:

There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

CHAPTER FOUR

REQUIREMENTS & DESIGN

This chapter illustrates the approaches taken to design an e-voting system. The chapter first addresses different types of requirements of the system. Then it discusses about the system design and gives an overview of the systems processes. Then, the database schema of the system is illustrated. Lastly, the user interface design has been developed.

4.1 Requirements Elicitation

The requirements of a system are characteristics of a system it needs to have. The requirements have been collected in the planning phase of the SDLC. Different kinds of data collection methods have been utilized to obtain the requirements of the system.

4.1.1 Functional Requirements

These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

- The system will be able to allow the user to create an account.
- Allow registered users to login to the system.
- The system should authenticate users during login,
  i.e. the user should be what he/ she claims to be.
- The system should allow an authenticated user to cast a vote.
- The system should send voting result to users.
- The system should allow the administrator (voting officer) to manage candidates, i.e. adding contestants, removing contestants

4.1.2 Non-Functional Requirements

These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioural requirements.

- ❑ System will support response times for addressing server issues in less time.
- ❑ System will provide documentation to inform users of system functionality and any charge to the system.
- ❑ The system will provide a friendly graphical Interface to ensure the case of use when end users utilize system functionality. (Usability)
- ❑ The system will be able to be integrated in the future if need be.
- ❑ The application will be available all the time for uses on google play to download.

4.2 Process Modelling

Process modelling is used in a project to depict the processes of data in an application. The e-voting application will be implemented using Model-View-Controller (MVC) design pattern. These processes are mostly implemented as business logic in application controllers. There are different tools for process modelling in SSADM (**Structured Systems Analysis and Design Method**). Context diagram and data flow diagram will be used to model the processes of the system.

4.2.2 Context Diagram

A context diagram outlines how external entities interact with an internal software system. It is primarily used to help businesses wrap their heads around the scope of a system. As a result, they can figure out how best to design a new system and its requirements or how to improve an existing system.

Context diagrams are high-level diagrams, meaning they don’t go into the detailed ins and outs of the system. Instead, they map out an entire system in a way that’s simple, clear, and easy to understand.

4.2.2.1 User activity

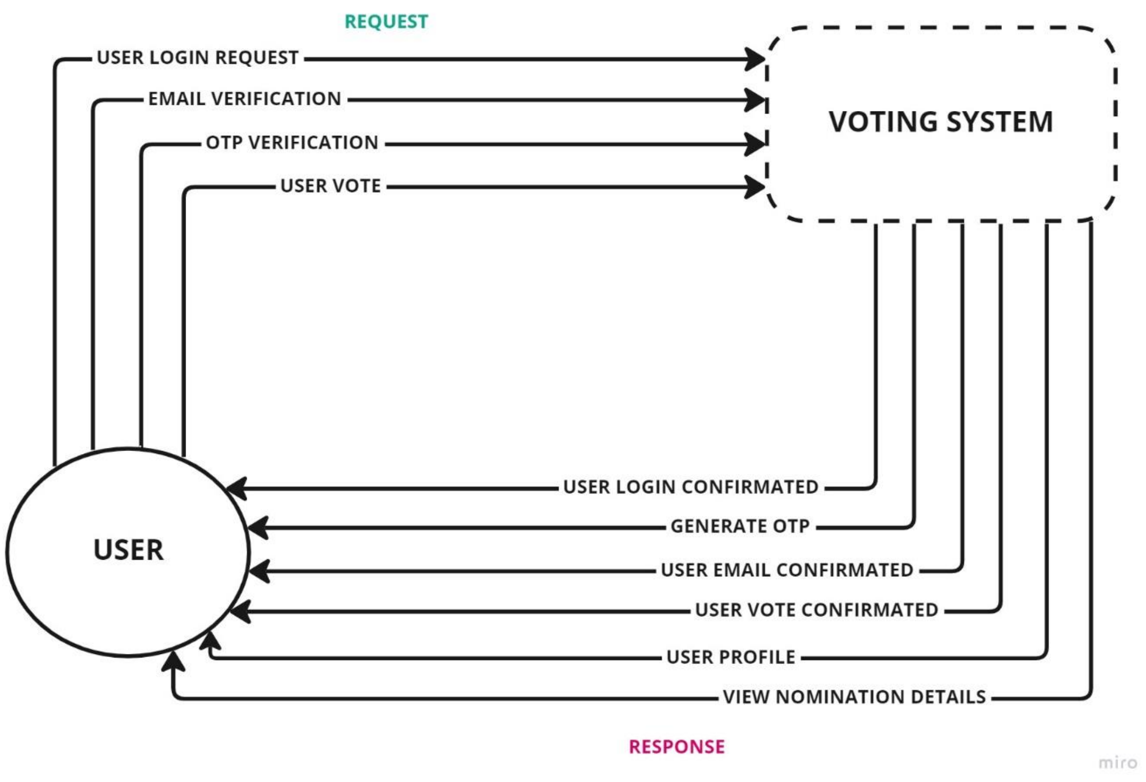

Figure 5: User context diagram

System Context Diagrams ... represent all users entities interacting with a system ... Such a diagram pictures the system at the center, with no details of its interior structure, surrounded by all its interacting systems, environments, and activities.

4.2.2.2 Admin Activity

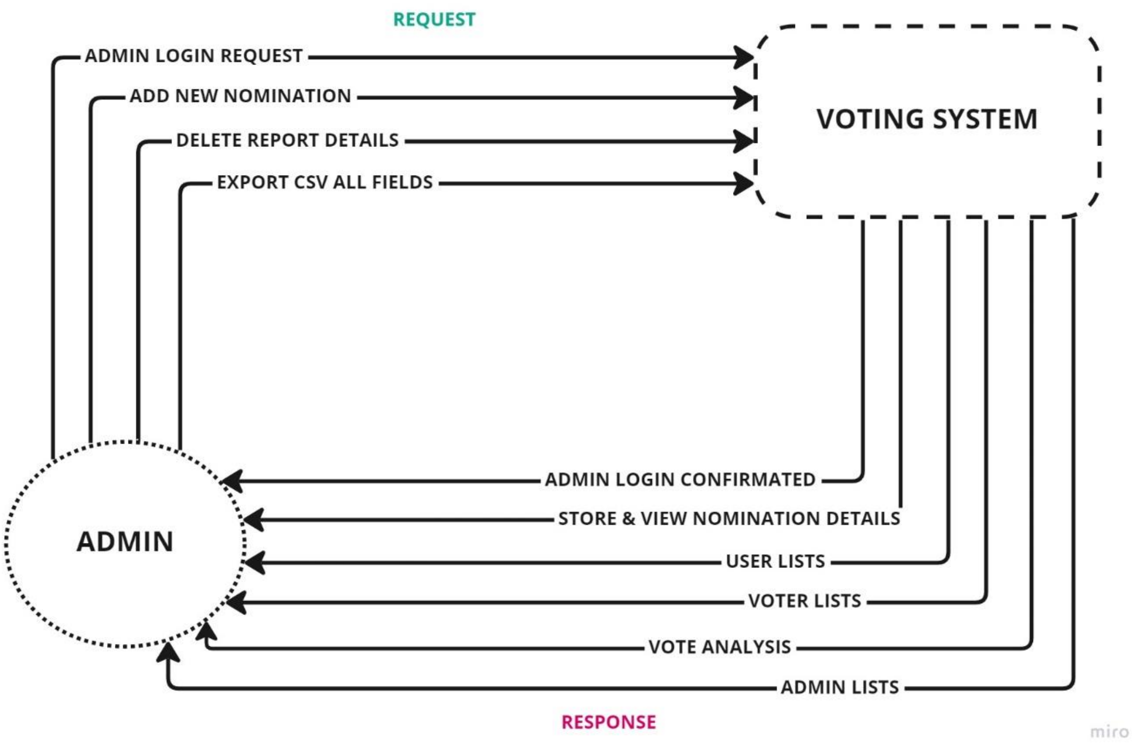

Figure 6: Admin context diagram

System Context Diagrams ... represent admin entities interacting with a system ... Such a diagram pictures the system at the center, with no details of its interior structure, surrounded by all its interacting systems, environments, and activities.

4.2.3 Data Flow Diagram

A data flow diagram (DFD) is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement. They are often elements of a format methodology such as Structured System Analysis and Design Method (SSADM). Superficially, DFDs can resemble flow charts or Unified Modeling Language (UML), but they are not meant to represent details of software logic.

DFDs can be hierarchically organized, which helps in progressively partitioning and analysing large systems. Such DFDs are called levelled DFDs. Context diagram is a diagram in which the entire system is treated as a single process and all its inputs, outputs, sinks, and sources are identified and shown.

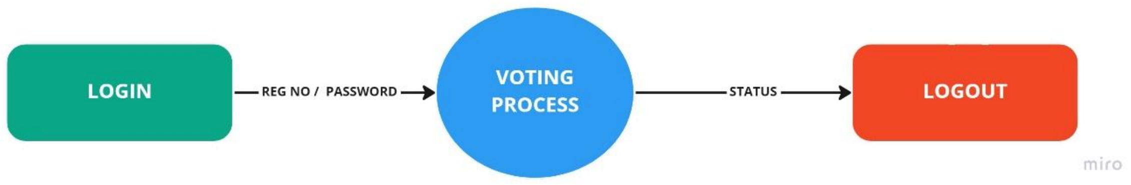

Figure 7: (a) level 1 DFD

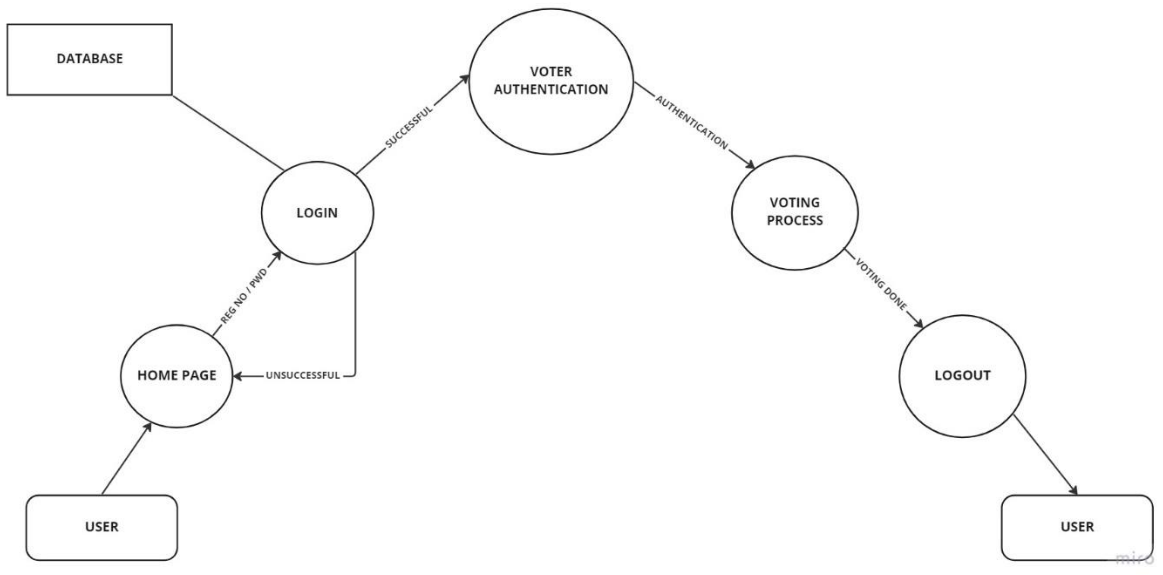

Figure 8: (b) level 2 DFD

4.3 Data Modelling

4.3.1 Entity Relationship Diagram (ERD)

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how "entities" such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

ER diagrams are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities instead of relationships between entities themselves. ER diagrams also are often used in conjunction with data flow diagrams (DFDs), which map out the flow of information for processes or systems.

User of entity relationship diagram

- ❑ **Database design:** ER diagrams are used to model and design relational databases, in terms of logic and business rules (in a logical data model) and in terms of the specific technology to be implemented (in a physical data model.) In software engineering, an ER diagram is often an initial step in determining requirements for an information systems project. It's also later used to model a particular database or databases. A relational database has an equivalent relational table and can potentially be expressed that way as needed.

- ❑ **Database troubleshooting:** ER diagrams are used to analyze existing databases to find and resolve problems in logic or deployment. Drawing the diagram should reveal where it's going wrong.

- **Business information systems:** The diagrams are used to design or analyze relational databases used in business processes. Any business process that uses fielded data involving entities, actions and interplay can potentially benefit from a relational database. It can streamline processes, uncover information more easily and improve results.

- **Business process re-engineering (BPR):** ER diagrams help in analyzing databases used in business process re-engineering and in modeling a new database setup. Education: Databases are today's method of storing relational information for educational purposes and later retrieval, so ER Diagrams can be valuable in planning those data structures.

- **Research:** Since so much research focuses on structured data, ER diagrams can play a key role in setting up useful databases to analyze the data.

Entity Relationship Diagram

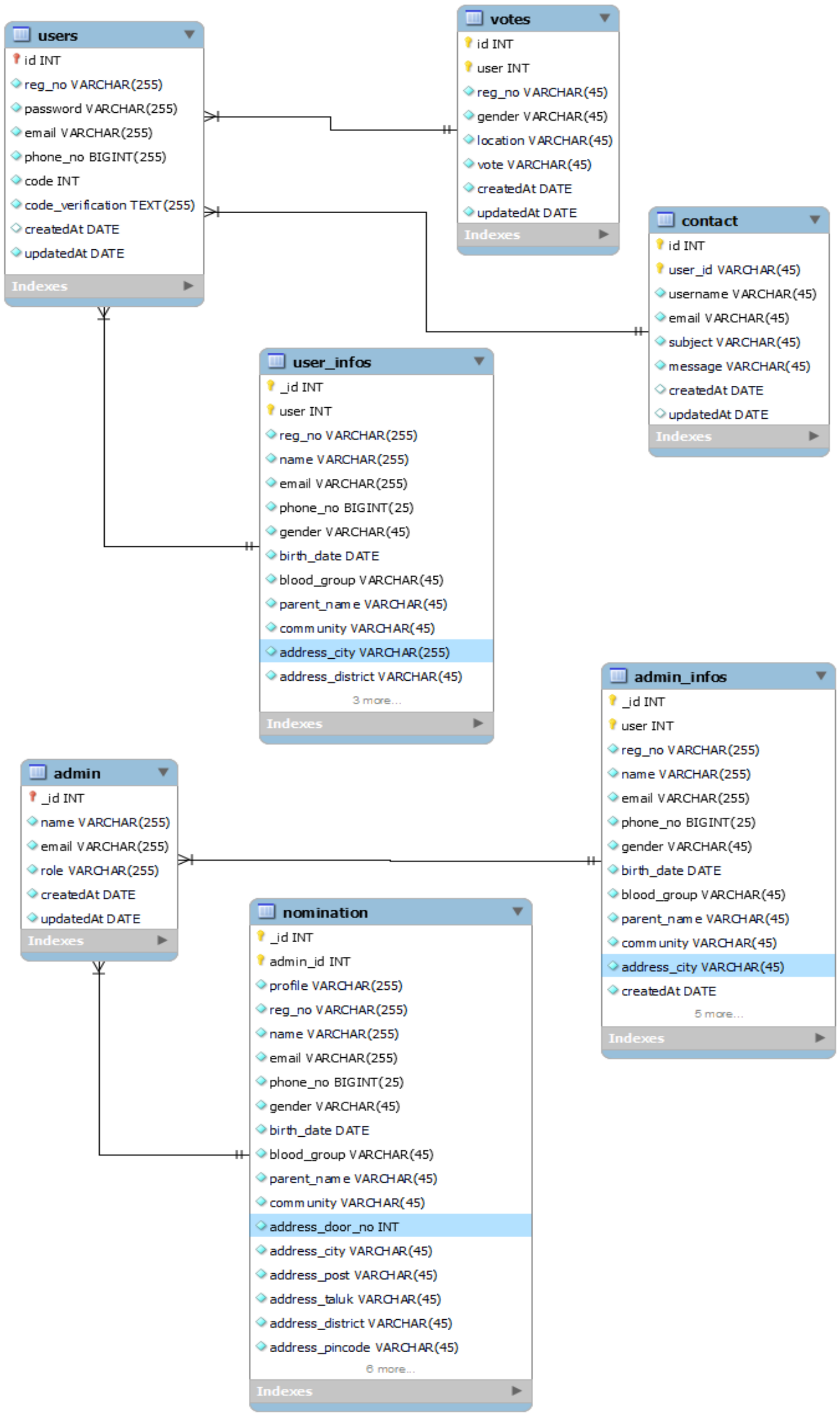

Figure 8: Entity relationship diagram of the system.

4.5 User Interface Design

As specified earlier, the user interface of the system should be responsive. So, the interface can adopt any screen size. As the system can be accessed by devices with large screen (e.g., desktop computer, smart television browser) as well as devices with small screen (e.g., smart phone, tablet), every page of the application is designed for two types of screen differently.

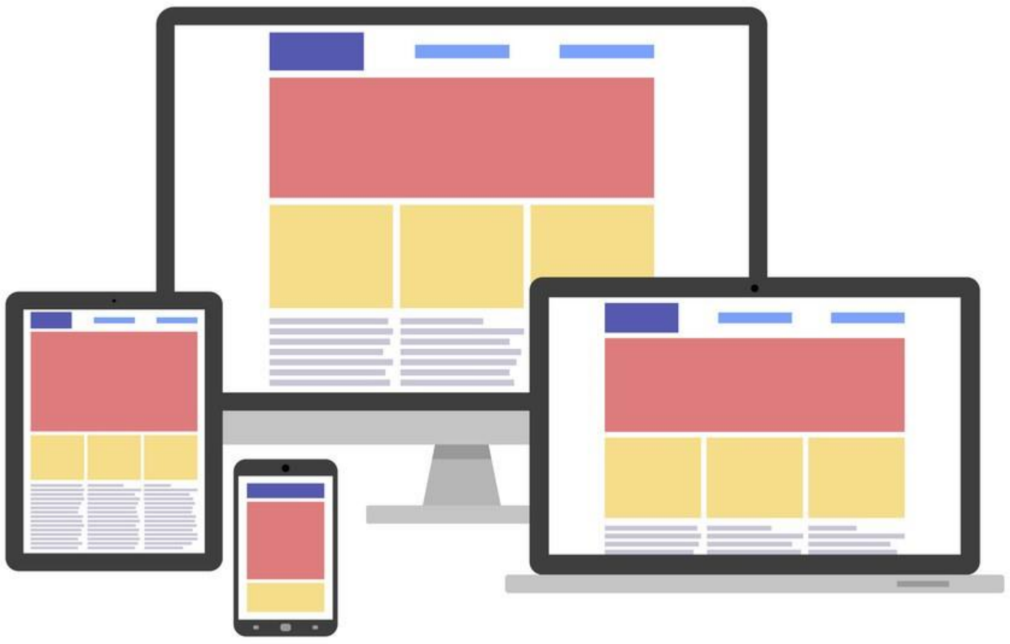

Figure 8: Difference of screen layouts for different size of screens.

4.5.1 User Details

User Home Screen

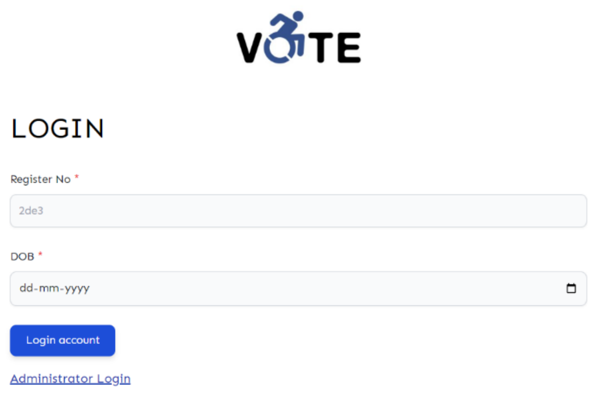

Figure 9: User login screen

User login with register number and DOB (password)