**Final Project Report**

**Web-based Online Voting System**



**Project Supervisor**

Kainat Malik

**Submitted By**

F220258033

Irum Zaidi VU ID: BC190409045

**Software Projects & Research Section,**

**Department of Computer Sciences,**

**Virtual University of Pakistan**



|  |
| --- |

**CERTIFICATE**

This is to certify that Irum Zaidi (BC190409045) has worked on and completed their Software Project at the Software & Research Projects Section, Department of Computer Sciences, Virtual University of Pakistan in partial fulfillment of the requirement for the degree of BS in Computer Sciences under my guidance and supervision.

In our opinion, it is satisfactory and up to the mark and therefore fulfills the requirements of a BS in Computer Sciences.

**Supervisor / Internal Examiner**

Kainat Malik

Supervisor,

Software Projects & Research Section,

Department of Computer Sciences

Virtual University of Pakistan

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Signature)

**External Examiner/Subject Specialist**

<<External Supervisor Name>>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Signature)

**Accepted by:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_**

(For office use)

**EXORDIUM**

**In the name of Allah, the Compassionate, the Merciful.**

**Praise be to Allah, Lord of Creation,**

**The Compassionate, the Merciful,**

**King of Judgment-day!**

**You alone we worship, and to You alone, we pray for help,**

**Guide us to the straight path**

**The path of those whom You have favored,**

**Not of those who have incurred Your wrath,**

**Nor of those who have gone astray.**

**DEDICATION**

I would like to dedicate this project to Allah Almighty, my strong pillar, source of wisdom and guidance, and then after it to my Parents, teachers, friends, and my project supervisor ***Kainat Malik***, who stood beside me and encouraged me to complete this project and gave me enough assistance to complete this.

This project would be very difficult if these personalities wouldn’t be there for me when I needed them the most.

**ACKNOWLEDGEMENT**

I would like to say my sincere gratitude to my *Parents* for their invaluable advice, ideas, and support in helping me finish the project successfully.

**PREFACE**

"**Gathering and Analysing Info**" is the first chapter, which also contains the SRS Document. A thorough description of the Use Cases, Usage Scenarios, Functional and Non-Functional Requirements, Methodology, Work Plan, and Gantt chart are included in its sub-parts, along with the Introduction, Scope of the Project, and Purpose.

"**Designing Project**" is the second chapter, which contains documents from the Design phase. Introduction, Scope and Purpose, Architectural Representation, Dynamic Model, Object Model, Deployment Model, Database Model, and Graphical User Interfaces are the sections of this chapter.

"**Development Plan**" is the final chapter, and it contains the final application, final report, and final presentation.

**TABLE OF CONTENTS**

|  |
| --- |

**CHAPTER NO. 1**

**gathering & Analyzing info 9**

1.1 Introduction ……………..……….……………..……….……………..……………..10

1.2 purpose ……………..……….……………..……..……….……………..……………..10

1.3 scope ……………..……….……………..……..……….……………..………………...10

1.4 definitions, acronyms, and abbreviations ...……………..……….………………..10

1.5 Project requirements ……………..……….……………..……..……….……………11

1.5.1 FUNCTIONAL REQUIREMENTS ……………..……….……………..……..……...11

1.5.2 Non-Functional Requirements ….…………..……….……………..……..……12

1.6 use cases and usage scenarios …………….……….……………..……..………..…13

1.6.1 Use Case Diagrams ….……………..……….……………..……..………….14

1.6.2 Usage Scenarios ….……………..……….……………..……..…………….14

1.7 Development Methodology ….……………...……….………...…..……..…………20

1.7.1 Chosen Methodology …..……………...………..……...…..……..……...23

1.7.2 Reasons for Chosen Methodology …..……………...……….……...……23

1.7.3 Work Plan (Gantt Chart) …..……………...………..……......……..……24

1.7.4 Project Schedule (Submission Calendar) …..……………...………..…...24

**CHAPTER NO. 2**

**designing project 25**

2.1 Introduction .……………...………..…....……………...………..…....……………...26

2.2 purpose .……………...………..…....……………...………..… ....… ....……………...26

2.3 scope ……………...………..…..……..……………...………..…..……..…..……...…27

2.4 definitions, acronyms, and abbreviations ……………...………..…..……..…..……..…27

2.5 Architectural Representation (Architecture Diagram) ……………...………..…..…27

2.6 Dynamic Model: Sequence Diagrams ……………...………..…..……………...…………28

2.7 Object Model/Logical Model: Class Diagram ……………...………..…..……….……..41

2.8 Database Model (Database Diagram) ……………...………..…..…………….....……..41

2.9 Graphical User Interfaces ……………...………..…..……………...………..…..……...42

**Chapter no.3**

**Development 49**

3.1 Development plan (Architecture Diagram) …………………………………….50

**CHAPTER 1**

Gathering & Analyzing Info

## Introduction

A Software Requirement Specification (SRS) is a comprehensive document that encompasses a detailed description of how a system will function. It outlines both Functional and non-functional requirements, along with a set of use cases that illustrate the user interactions the software must support. The SRS also incorporates usage scenarios and the adopted methodology for the development process.

To be effective, an SRS should meet several criteria. It must be complete, ensuring that all essential aspects of the system are addressed. It should be consistent, ensuring that there are no conflicting or contradictory statements. The document should be verifiable, allowing its requirements to be tested and validated. An SRS should also be unambiguous, leaving no room for misinterpretation or confusion. Lastly, it must be correct, accurately representing the intended functionality and behavior of the software.

## Purpose

The SRS document should have a well-defined purpose, clearly stated to aid the reader in comprehending its objective and potential applications.

## Scope

The SRS document should clearly define the scope of the project, outlining what aspects are included and what is excluded. In this case, the project at hand is the Web-Based Online Voting System, designed to enable users to cast votes and view results online. The system is web-based, taking advantage of the widespread availability of the internet, computers, and mobile devices, making it convenient for users to cast their ballots from anywhere with just a click. The primary aim of this project is to save human lives and ensure accessibility and participation in the voting process, even under unfavorable circumstances.

## Definitions, Acronyms, and Abbreviations

**Definitions**

* **Web Application:** A computer program that can be accessed by a web interface or web browser, usually installed on a server, and providing users with material or functionality over the internet.
* **User Interface (UI):** A web application's visual and interactive features, such as forms, screens, menus, buttons, and other graphical elements, that allow users to engage with the system.
* **Front-End:** The client-side element of a web application that manages display and user interaction and executes on the user's device (which could be a browser).
* **Back-End:** The server-side element of an online application that handles user requests, does calculations, and communicates with databases or other systems to supply the front end with data and functionality.
* **Database:** A systematic collection of data that has been organized and saved to allow for quick access, editing, and administration.

**Acronyms, and Abbreviations**

* **HTML (Hypertext Markup Language):** HTML is a markup language used to design web page layouts and contents.
* **CSS (Cascading Style Sheets):** A language for creating style sheets that describe the layout, colors, and fonts that make up a web page's visual look.
* **JavaScript:** A popular language for client-side scripting that allows dynamic and interactive elements on websites.
* **UX (User Experience):** The whole interaction and pleasure that a user has with a web program; this includes elements like usability, effectiveness, and enjoyment.
* **HTTP (Hypertext Transfer Protocol):** This fundamental protocol governs how messages are structured and conveyed between web servers and clients when data is transferred over the internet.
* **URL (Uniform Resource Locator):** The address of a website that indicates where a resource is located on the internet and is often used for accessing web pages or other sources of information.
* **WBOVS (Web-Based Online Voting System):** It is an online voting project that will conduct ballots online.

## Project requirements

### Functional Requirements

Functional requirements define the specific functionalities and capabilities that the web-based online voting system must possess to fulfill its purpose effectively. Here are the functional requirements for the online voting system:

**User Registration:**

* The system should allow individuals to register as voters by providing essential information, such as name, address, age, and identification details.
* The registration process should include validation and verification steps to ensure the accuracy of voter information.

**User Authentication:**

* The system must provide secure login functionality for users to access their accounts and cast their votes.
* User authentication should involve a username/password or other secure authentication mechanisms.

**Profile Management:**

* Registered users should be able to update their profile information, such as personal details if required.

**Voter Approval:**

* The system should have an approval functionality where registered voters' information is reviewed and approved by the admin to become eligible for voting.

**Voter ID Generation:**

* Once approved, the system should generate a unique Voter ID for each eligible voter, which will be used for vote casting and verification.

**Voting Process:**

* The system shall provide a user-friendly interface for voters to cast their votes for their preferred candidates.
* Each user can cast only one vote per position.

**Candidate Registration:**

* The admin can register candidates for the election by providing their personal information and details about the position they are running for.

**Candidate Management:**

* The admin shall have the authority to add new candidates to the election or remove candidates when necessary.
* The admin shall have the ability to create, modify, or remove positions for which users can cast their votes.

**Election Positions Management:**

* The admin should be able to define and manage the positions available for voting in the election.
* The admin can add, remove and edit voting positions.

**Vote Counting:**

* The system should accurately count the votes cast for each candidate and position.

**Election Results:**

* The system shall display the election results to both admin and voters securely and transparently. The admin has the authority to display or hide results from the user.

**Election Statistics:**

* The system shall display real-time statistics on the total number of registered voters, candidates, and available positions for voting.

**Logout Functionality:**

* The system shall provide a secure logout mechanism for users and admins to exit the system after completing their tasks.

These functional requirements form the foundation of the online voting system, enabling it to facilitate a fair, secure, and efficient election process for all stakeholders involved.

By redefining these functional requirements, the online voting system can be better specified and developed to ensure smooth user experiences, enhanced security, and efficient management of the election process.

### Non-Functional Requirements

Non-functional requirements are critical aspects of a web-based online voting system that define its overall performance, usability, security, and other essential qualities. Here are some non-functional requirements specific to an online voting system:

**Security:**

* The system must implement robust security measures to protect against unauthorized access, data breaches, and tampering of votes.
* Use of encryption to ensure secure data transmission between clients and servers.
* Implement measures to prevent and detect fraudulent activities, such as multiple voting or vote manipulation.

**Reliability:**

* The system must be available and accessible during the voting period, with minimal downtime.
* It must be capable of supporting several concurrent users without experiencing performance deterioration or system breakdown.
* Data integrity must be maintained, and votes should be accurately recorded and counted.

**Scalability:**

* The system should be able to accommodate an increasing number of voters and data as the user base grows.
* Scaling should not compromise system performance.

**Usability:**

* The interface should be user-friendly and intuitive, catering to voters of varying technical expertise.
* Clear instructions should be provided to guide users through the voting process.
* The system should support accessibility standards to ensure inclusivity for users with disabilities.

**Performance:**

* The system must respond quickly to user interactions, providing a seamless voting experience.
* Minimal latency should be ensured to avoid frustration among voters.

**Compatibility:**

* The web-based system should be compatible with different browsers and operating systems, ensuring broad accessibility for users.

**Privacy:**

* Voter information and choices must be confidential and not linked to individuals.
* The system should comply with relevant data protection and privacy laws.

## Use Cases and Usage Scenarios

Use cases and usage scenarios are integral components of software development, offering detailed insights into user interactions and experiences. By understanding the significance of these elements, developers and stakeholders can craft applications that align with user needs and expectations. Use cases focus on detailed interactions between users and the system, while user stories provide brief descriptions of user requirements.

### Use Case Diagram

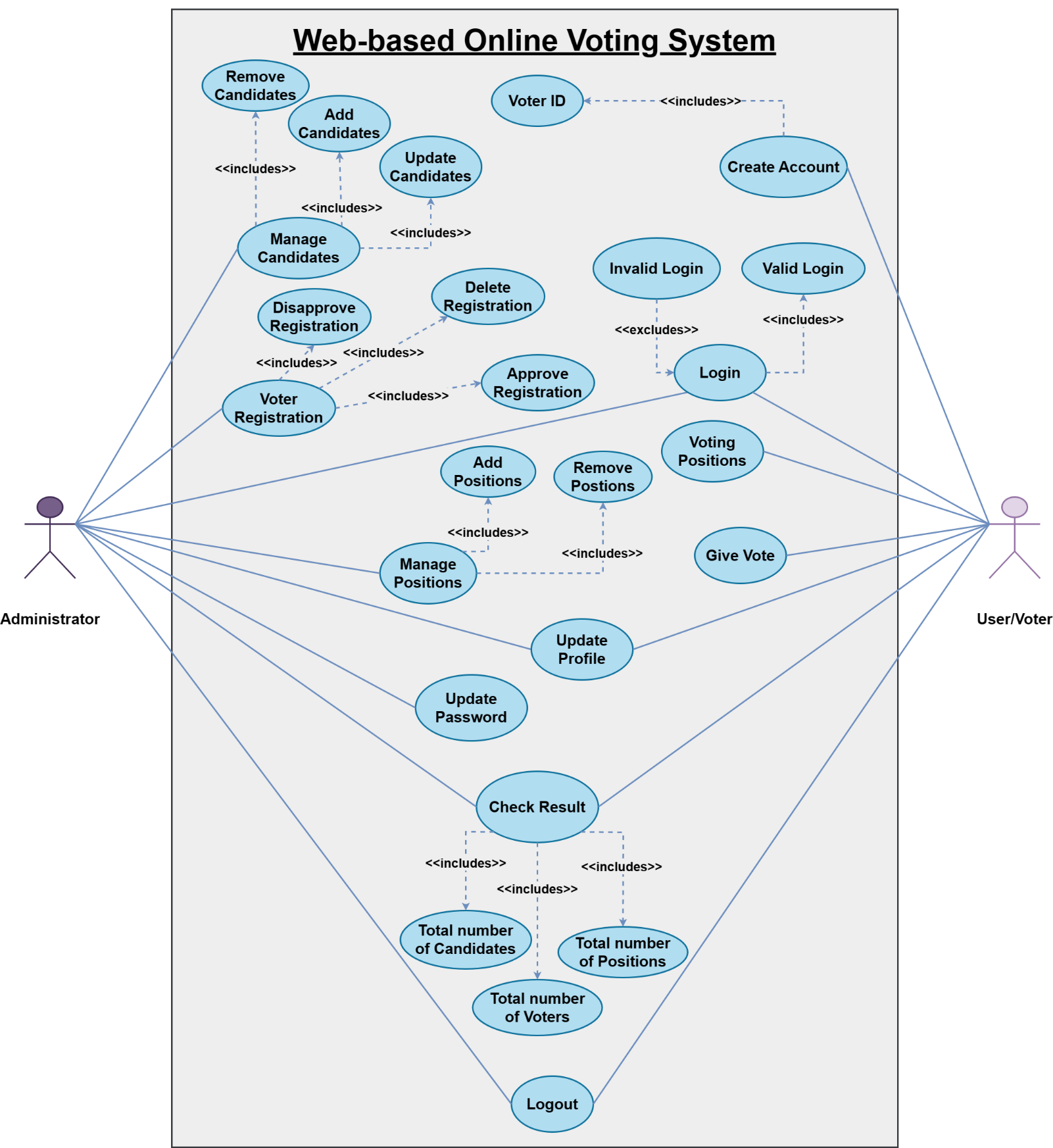


Figure 1.6.1.1: Use Case Diagram.

### Usage Scenarios

#### **Voter Module**

Registration:

| **Use Case Title:** Create Account. | **Use Case ID:** V-01. |
| --- | --- |
| **Action:** The voter creates an account to register. | **Author:** BC190409045 |
| **Description:** To use the system, the voters must register to the system. This explains the registration process. | |
| **Precondition:** TheInternet connection must be stable. | |
| **Post-Condition:** A user account will be created. | |
|  | |
| **Normal Course of Events:**   1. The voter enters the homepage. 2. He Clicks on the “Registration” page. 3. The system prompts the application form. 4. The User fills in the necessary information related to him in the application form. 5. He sends the request for registration by using the “send” button. | |
|  | |
| **Alternate Path:** If the user does not click the “Submit” button but clicks the cancel button, the system will display the home page again.  If the user does not provide the required information, the form will not proceed. | |
| **Exceptions:** At any stage, the server may disconnect or have any internet issues, then the system will not save the data on the server. | |

Login:

| **Use Case Title:** Login. | **Use Case ID:** V-02. |
| --- | --- |
| **Action:** Voters can log in to the system. | **Author:** BC190409045. |
| **Description:** The voter can log in to the system. | |
| **Precondition:** The Internet connection should be stable. | |
| **Post-Condition:** Voter login successfully into the system. | |
|  | |
| **Normal Course of Events:**   * The voter enters their email and password. * Then click on the login button. | |
|  | |
| **Alternate Path:** The voter entered an invalid password or email. | |
| **Exceptions:** Fields could be empty. The voter entered an invalid email or password. | |

Vote Casting:

| **Use Case Title:** Give Vote. | **Use Case ID:** V-03. |
| --- | --- |
| **Action:** Voters can cast their vote. | **Author:** BC190409045. |
| **Description:** The user can cast their vote for the candidate they want. | |
| **Precondition:** The voter must be logged in to the system. | |
| **Post-Condition:** Voter cast their vote. | |
|  | |
| **Normal Course of Events:** | |
| * The voter logged in to the system. * Go to the Current Vote page. * Choose a candidate for the vote. * Then, cast their vote. | |
|  | |
| **Alternate Path:** The voter may click on the position menu instead of the voting menu | |
| **Exceptions:** Slow server connection or internet connection. | |

Check Result:

| **Use Case Title:** Check Result | **Use Case ID:** V-04. |
| --- | --- |
| **Action:** The voter wants to see the voting results. | **Author:** BC190409045. |
| **Description:** This describes the process of how voters view the election results by using the system. | |
| **Precondition:** The voter must be logged in to the system. | |
| **Post-Condition:** The voting result would be displayed. | |
|  | |
| **Normal Course of Events:** | |
| * Voter clicks on the Poll Result link. If the admin wants to show the result, then the result will be displayed to the voter, otherwise, the result will not be displayed. * The system displays the voting results. | |
|  | |
| **Alternate Path:** The user may click on another menu instead of the result page. | |
| **Exceptions:** Slow server connection or internet connection. | |

Profile:

| **Use Case Title:** Update Profile. | **Use Case ID:** V-05. |
| --- | --- |
| **Action:** Voters can update information about themselves. | **Author:** BC190409045 |
| **Description:** How voters can update their profile. | |
| **Precondition:** The voter must be already registered. | |
| **Post-Condition:** The information will be updated. | |
|  | |
| **Normal Course of Events:**   1. Voter clicks on the profile icon. 2. Voter updates the profile and clicks on the “Save” button. 3. The system will update the profile of the user if all the information is complete. | |
|  | |
| **Alternate Path:** If the information is not complete, then the system will not update the information on the server. | |
| **Exceptions:** Loss of connection will cause an error. | |

Logout:

| **Use Case Title:** Logout. | **Use Case ID:** V-06. |
| --- | --- |
| **Action:** The voter can log out of the system. | **Author:** BC190409045. |
| **Description:** The voter can log out of the system. | |
| **Precondition:** The voter must be already logged in to the system.The Internet connection should be stable. | |
| **Post-Condition:** The user logs out successfully from the system. | |
|  | |
| **Normal Course of Events:** | |
| * The user will click on the logout button. * The login page would be displayed on the screen. | |
|  | |
| **Alternate Path:** The voter clicks on another menu instead of logout. | |
| **Exceptions:** Unstable internet connection. | |

#### **Admin Module**

Login:

| **Use Case Title:** Login. | **Use Case ID:** A-01. |
| --- | --- |
| **Action:** The administrator can log in to the system. | **Author:** BC190409045. |
| **Description:** The administrator can log in to the system. | |
| **Precondition:** The Internet connection should be stable. | |
| **Post-Condition:** Admin login successfully into the system. | |
|  | |
| **Normal Course of Events:**   * The admin enters their email and password. * Then click on the login button. | |
|  | |
| **Alternate Path:** The admin entered an invalid password or email. | |
| **Exceptions:** Fields could be empty. The admin entered an invalid email or password. | |

Manage Position:

| **Use Case Title:** Manage Positions. | **Use Case ID:** A-02. |
| --- | --- |
| **Action:** The administrator can manage the candidates’ positions. | **Author:** BC190409045. |
| **Description:** How administrator manages candidate’s positions in election. | |
| **Precondition:** The candidate must be already in the system. | |
| **Post-Condition:** Candidate position after the election. | |
|  | |
| **Normal Course of Events:** | |
| * Administrator login to the system. * Add position into the system for voting. * Click the save button. * The admin can also update the positions. | |
|  | |
| **Alternate Path:** Maybe the administrator presses the “Cancel” button instead of “Publish”. | |
| **Exceptions:** Server connection may lose. | |

Manage Candidate:

| **Use Case Title:** Manage Candidates. | **Use Case ID:** A-03. |
| --- | --- |
| **Action:** The administrator can manage candidates. | **Author:** BC190409045. |
| **Description:** How the administrator manages candidates for election. | |
| **Precondition:** The server must be connected. The position must be added to the system. | |
| **Post-Condition:** Candidate added for voting. | |
|  | |
| **Normal Course of Events:** | |
| * Administrator login to the system. * Creates candidate. * Enter the candidate’s data. * Save the candidate’s information. * Candidate added to the system. * The admin can also update the candidate profile. | |
|  | |
| **Alternate Path:** Maybe the administrator presses the “Cancel” button instead of “Save”. | |
| **Exceptions:** Fields could be empty. The administrator entered invalid data. | |

Approve Registration:

| **Use Case Title:** Approve Registration. | **Use Case ID:** A-04. |
| --- | --- |
| **Action:** The administrator approves the voter registration. | **Author:** BC190409045. |
| **Description:** This describes how the administrator will approve the voter registration and generate a Voter ID for the voter. | |
| **Precondition:** Voter registration requests must be there. | |
| **Post-Condition:** The voter request for registration will proceed accordingly. | |
|  | |
| **Normal Course of Events:**   * Administrator selects the Voters page. * Administrator can read Voter information. * If the information is okay. Then admin can approve the request. Otherwise, the admin can disapprove and delete it. * When the admin approves a voter registration, a Voter ID is generated for that voter. | |
|  | |
| **Alternate Path:** If the given information is not correct, the administrator will not approve the voter registration. | |
| **Exceptions:** Incomplete information and loss of internet will not save data on the server. | |

Result:

| **Use Case Title:** Result | **Use Case ID:** A-05. |
| --- | --- |
| **Action:** The administrator wants to see and publish the voting results. | **Author:** BC190409045. |
| **Description:** This describes the process of how administrators view and publish the election results by using the system. | |
| **Precondition:** The admin must be logged in to the system. | |
| **Post-Condition:** The voting result would be displayed. | |
|  | |
| **Normal Course of Events:** | |
| * Admin clicks on the election result link. * He chooses a position and presses clicks on the button “show results”. * The system displays the required information according to the selected choices. | |
|  | |
| **Alternate Path:** The admin may click on another menu instead of the result page. | |
| **Exceptions:** Slow server connection or internet connection. | |

Profile:

| **Use Case Title:** Update Profile. | **Use Case ID:** A-06. |
| --- | --- |
| **Action:** The administrator updates their profile. | **Author:** BC190409045 |
| **Description:** How administrator can update profile. | |
| **Precondition:** The admin must be logged in. | |
| **Post-Condition:** The information will be updated. | |
|  | |
| **Normal Course of Events:**   1. Administrator their profile to update. 2. Administrator updates the profile and clicks on the “Save” button. 3. The system will update the profile of the admin if all the information is complete. | |
|  | |
| **Alternate Path:** If the information is not complete, then the system will not update the information on the server. | |
| **Exceptions:** Loss of connection will cause an error. | |

Logout:

| **Use Case Title:** Logout. | **Use Case ID:** A-07. |
| --- | --- |
| **Action:** The administrator can log out of the system. | **Author:** BC190409045. |
| **Description:** The administrator can log out of the system. | |
| **Precondition:** The admin must be already logged in to the system. The Internet connection should be stable. | |
| **Post-Condition:** The admin logs out successfully from the system. | |
|  | |
| **Normal Course of Events:** | |
| * The admin will click on the logout button. * The login page would be displayed on the screen. | |
|  | |
| **Alternate Path:** Admin clicks on another menu instead of logout. | |
| **Exceptions:** Unstable internet connection. | |

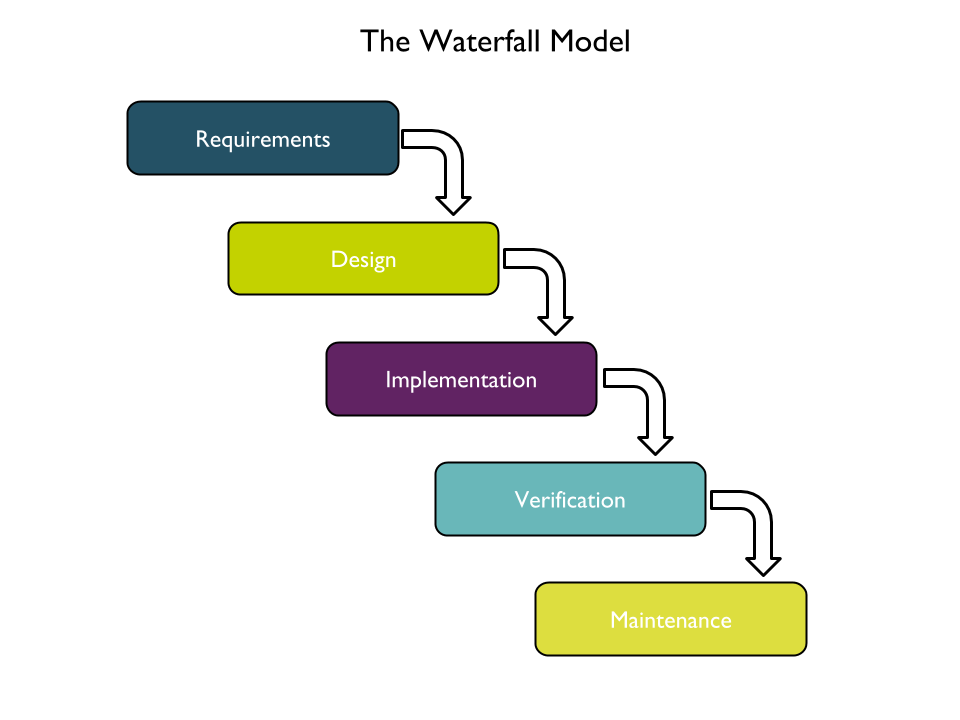
## Development Methodology

**Waterfall Model:**

Other engineering methods served as the foundation for the first published model of the software development process. The first Process Model to be introduced was the Waterfall Model. This model is referred to as the waterfall model because of the way that the phases cascade into one another. The waterfall model is also known as a linear-sequence life cycle model since it was the first SDLC strategy to be utilized for the software development process in a linear sequential flow.

Both understanding and using it are quite straightforward. Each step in a waterfall model must be finished before the subsequent phase may start. This implies that a phase of development can only start if the one before it is finished. Phases do not cross over into one another in the waterfall model, or they do not overlap. The requirements phase of the waterfall model is the first step, and the acceptance phase is where it ends.

The waterfall model is **documentation-driven**. As a result, it produces thorough and detailed documentation and makes maintenance considerably simpler. However, it has the drawback that any flaws in the required specifications are not found until the product is provided to the client after completion since client input is only collected when the product is eventually delivered. This has significant time and financial ramifications.

  
Figure: 1.7.1: Waterfall Model.

**Requirement Gathering and Analysis:**

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

**System Design:**

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

**Implementation:**

With inputs from system design, the system is first developed in small programs called units, which are integrated into 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 each unit. Post integration, the entire system is tested for any faults and failures.

**Deployment of the system:**

Once the functional and nonfunctional testing is done, the application will be deployed in the user environment.

**Maintenance:**

Some issues come up in the user environment. To fix those issues patches are released. Maintenance is done to deliver these changes in the user’s environment.

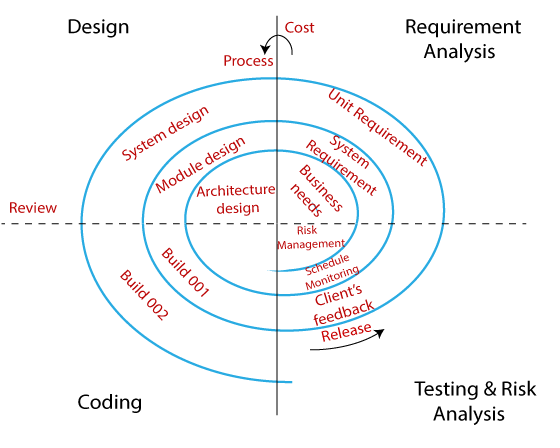
Now we will discuss the spiral model and the Vu model.

**Spiral Model:**

This model was developed by Barry Boehm. The main idea of this model is to avert risk as there is always an element of risk in the development of software. For example, key personnel may resign at a critical juncture, the manufacturer of the software development may go bankrupt, etc.

In its simplified form, the Spiral model is the Waterfall model plus risk analysis. In this case, each stage is preceded by the identification of alternatives and risk analysis and is then followed by evaluation and planning for the next phase. If risks cannot be resolved, the project is immediately terminated.

The main strength of the Spiral Model comes from the fact that it is very *sensitive* to risk. Because of the spiral nature of development, it is easy to *judge* how much to test and there is no distinction between development and maintenance. Each phase in the Spiral model starts with a *design goal* and ends with a *client reviewing* the process.

  
Figure: 1.7.2.: Spiral Model.

### Chosen Methodology

**VU Process Model:**

It is a synthesis of the waterfall and spiral methodologies. A **hybrid approach** to system development is another name for it. It consists of five stages, including requirement gathering and analysis, planning, analysis and design, development, and final report. Throughout discussions with the user, the system's objectives, services, and restrictions are decided upon during the requirement phase. The planning stage involves a coordinated effort to identify all possible challenges and create a solid plan to address each one. This stage often involves a lot of brainstorming between the client and the software development team. It lowers risks and faults while maximizing the system's quality.

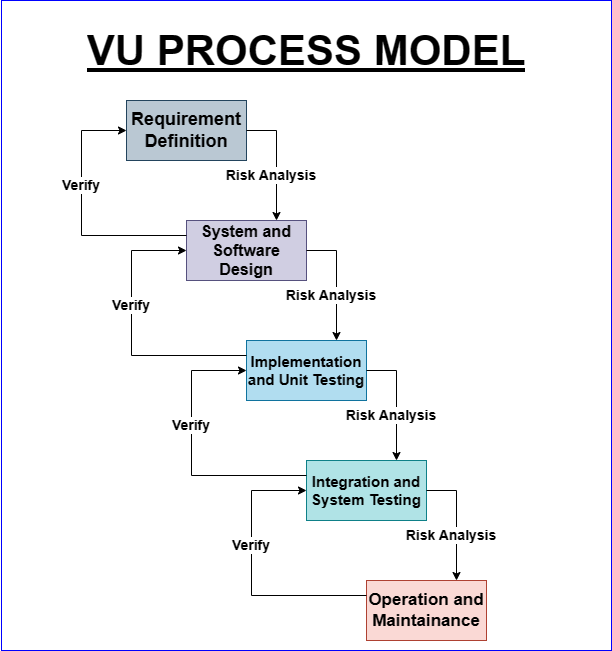


Figure 1.7.1.1: Process Model.

### Reasons for Chosen Methodology

The basic reasons for choosing VU Process Model for our project are: Our project is broken up into many parts, each of which must be finished in order and submitted to our supervisor. If errors are discovered, the supervisor recommends that we should fix them to enhance our project. Due to its spiral nature, this procedure will be used. We shall move on to the following step once the current one has been properly prepared and approved by our supervisor. Due to the VU Process model's waterfall structure, this will be done. As a result, both models are suppressed in one form, which will serve as our VU process model. The orderly completion of each stage will produce outcomes free of errors.

### Work Plan (Gantt Chart)

Here’s the Gantt Chart for the project:

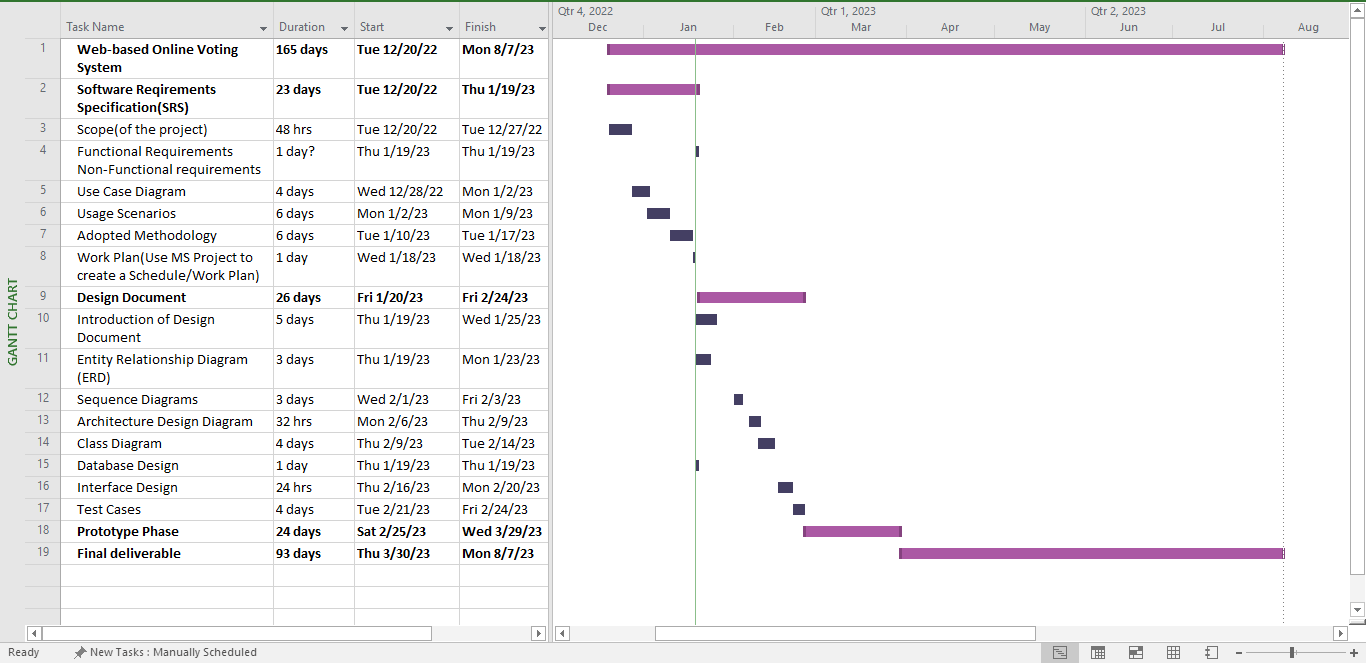


Figure 1.7.3.1: Gantt Chart

### Project Schedule (Submission Calendar)

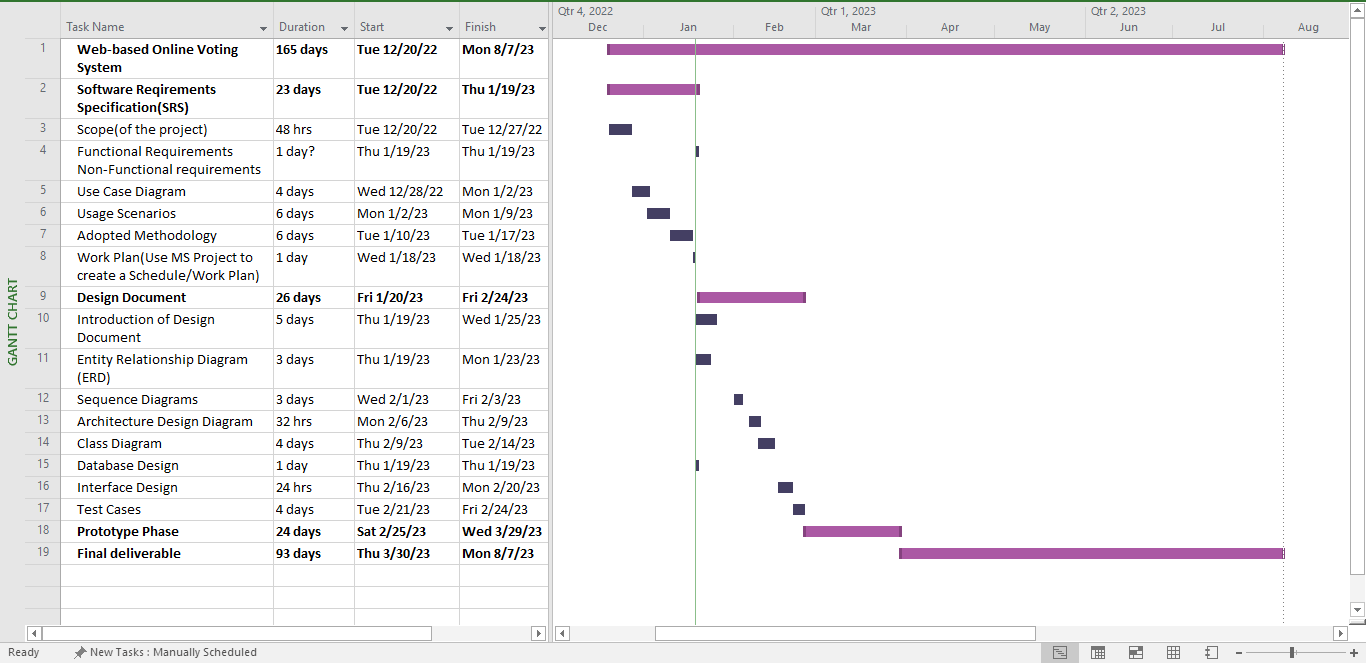


Figure 1.7.4.1: Submission Calendar

**CHAPTER 2**

Designing the Project

## 2.1. Introduction

The design document is an in-depth document that details the structure and requirements for a system, product, or project. A typical example would be an entity relationship diagram, sequence diagram, architecture design diagram, class diagram, database design, interface design, and test cases. It might also include information on the general structure, user interface, and features of the product. Stakeholders, designers, and developers make use of the design document as a template for creating and implementing the product to make sure it satisfies the project's needs and objectives. Additionally, it serves as a manual for the product's testing, upkeep, and future growth.

## 2.2. Purpose

The design document serves as a documented representation of a system's or product's design. It is used to explain the design to various people, including clients, executives, and developers. Systems or products may be planned, developed, and maintained using design papers.

The following categories frequently appear in design documents:

* **Introduction:** This part gives a general description of the system or product, outlining its goals, criteria, and intended customer.
* **System architecture:** this section describes the components, user interfaces, and data operations of the system as well as its general structure.
* **Design of the user interface:** This section discusses the system's overall appearance, including the buttons, menus, and other graphical features.
* **User experience design:** This part outlines how users will be interacting with the system, including the way they will use it to browse, input data, and complete activities.
* **Technical details:** This section contains comprehensive technical details on the system, including its programming languages, hardware specs, and database design.

Text, diagrams, and tables are just a few of the several writing styles that may be used in design documentation. The design document's structure should be customized to the project's unique requirements.

A crucial step in the software development process is the creation of design papers. They aid in ensuring that the system or end product is understood by all parties participating in the project. The overall performance of the system or product may be increased and faults can be avoided with the use of design papers.

The following are some advantages of utilizing design documents:

* **Better communication:** Design documents provide better communication among all parties involved, including clients, superiors, and developers. By doing so, mistakes may be avoided and a greater knowledge of the system or product may result.
* **Better quality:** By offering a thorough outline of the system's or product's development, design papers can contribute to an improvement in the system's or product's quality. This can aid in ensuring that the system or product is error-free and satisfies the demands of its consumers.
* **Lower expenses:** Design papers can help lower costs by minimizing mistakes and by making system or product modifications simpler.
* **Greater efficiency:** Design papers can contribute to greater efficiency by laying out a clear development strategy for the system or product. This may increase the likelihood that the project will be completed on time and under budget.

## 2.3. Scope

The process of integrating software requirements into an illustration of the software features and data required for the software's execution stage is known as software design. The software design document outlines the organizational structure that will be used to meet the requirements. It is the most crucial recommendation for code development, thus it must include all the details a programmer needs to produce code in the system. The goal of the design document is to give a detailed enough description of a system's architecture so that software development may move forward knowing exactly what needs to be created and how it should be produced.

## 2.4. Definitions, acronyms, and abbreviations

**Definitions:**

* **Web application:** A software program offering interactive features and functions through the internet that may be accessed using a web browser or web-enabled device.
* **User Interface (UI):** A software application's graphical and interactive features that users utilize to carry out tasks and retrieve information.
* **User Experience (UX):** A user's total impression of and pleasure with an online program, including its usability, effectiveness, and pleasant aspect.
* **Responsive Design:** The process of creating a web application that can adjust to multiple screen sizes and devices, including computers, tablets, and mobile phones, and offer an ideal user experience.

**Acronyms, and Abbreviations**

* **HTML:** Hypertext Markup Language.
* **CSS:** Cascading Style Sheets.
* **Bootstrap:** Bootstrap CSS Framework.
* **JS:** JavaScript.
* **UI/UX:** User Interface/User Experience.
* **DBMS:** Database Management System.
* **SQA:** Software Quality Assurance.
* **DD:** Design Document.

## 2.5. Architectural Representation (Architecture Diagram)

Architectural representation is the backbone of the architectural design process. Architectural representation is essential as it visually communicates complex design concepts to clients and stakeholders, ensuring a shared understanding of the project. Through various types of representation, architects can effectively communicate their ideas and ensure a shared vision with clients and stakeholders. As technology continues to evolve, the field of architectural representation will undoubtedly see further improvements, enhancing the way we bring architectural visions to life. Here’s the architectural representation of our Web-based Online Voting System.

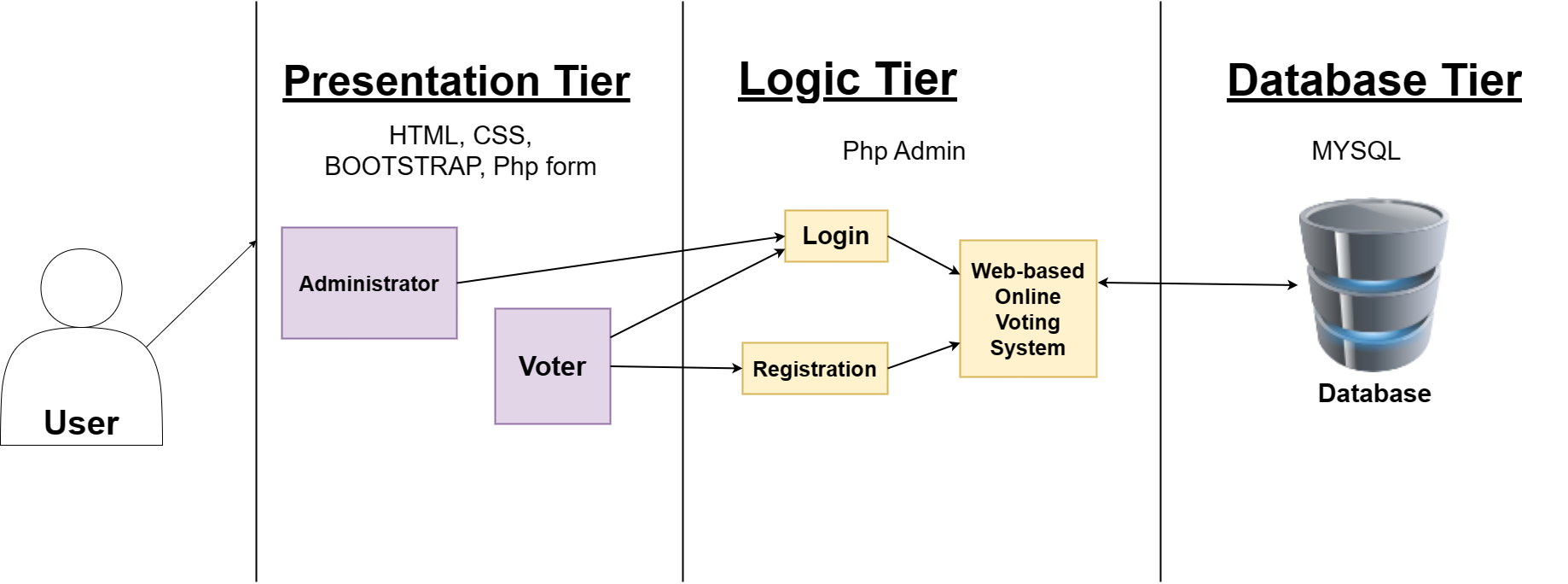


Figure 2.5.1.: Architecture Diagram.

## 2.6. Dynamic Model: Sequence Diagrams

Sequence diagrams are powerful tools that play a crucial role in the dynamic model of software development. They offer a visual representation of interactions, making it easier to understand complex systems. By creating effective sequence diagrams and following best practices, developers can improve collaboration, identify flaws, and enhance the overall software development process. Here are the Sequence Diagrams from the Web-Based Online Voting System

### 2.6.1. Admin Module

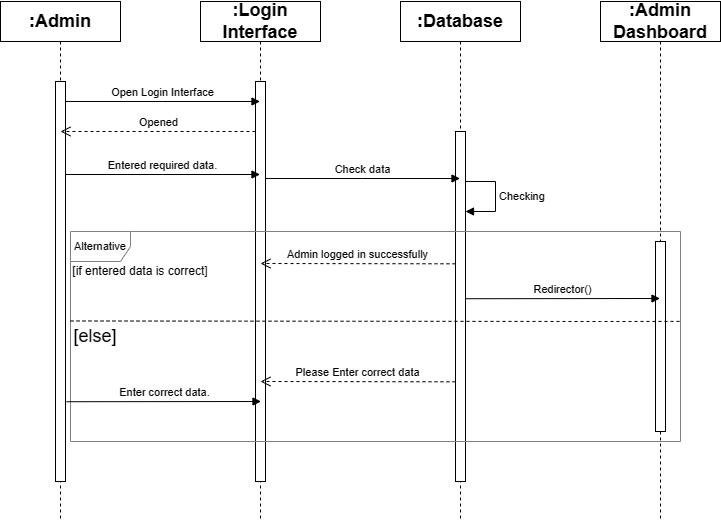


Figure 2.6.1.1.: Admin Login.

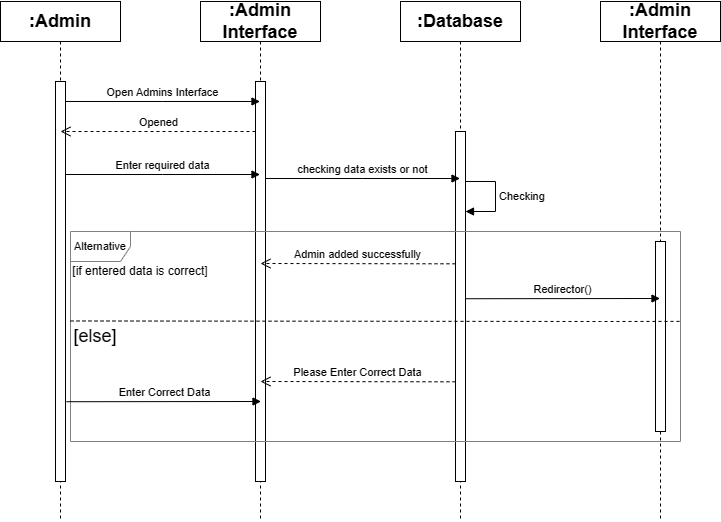


Figure 2.6.1.2.: Admin Added.

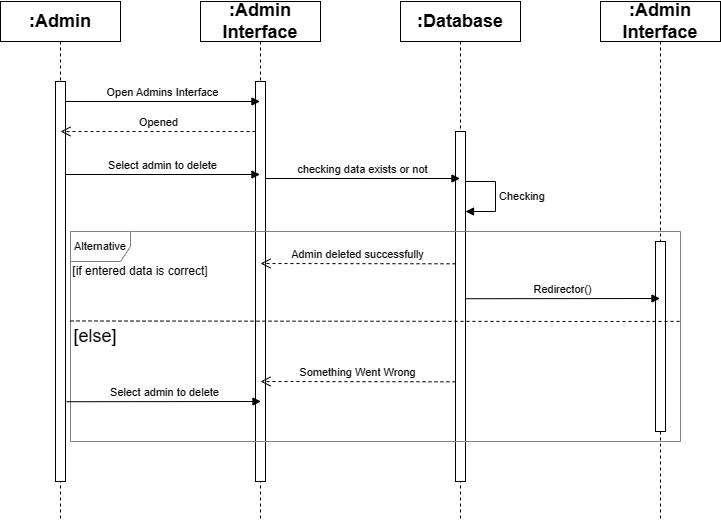


Figure 2.6.1.3.: Admin Deleted.

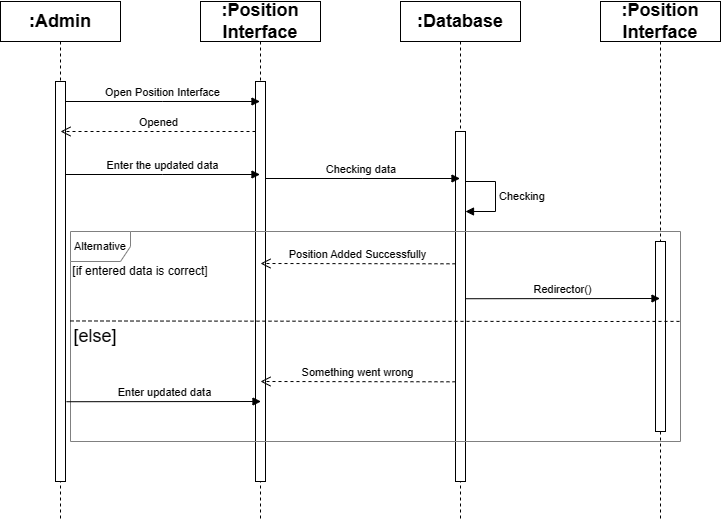


Figure 2.6.1.4.: Position Added.

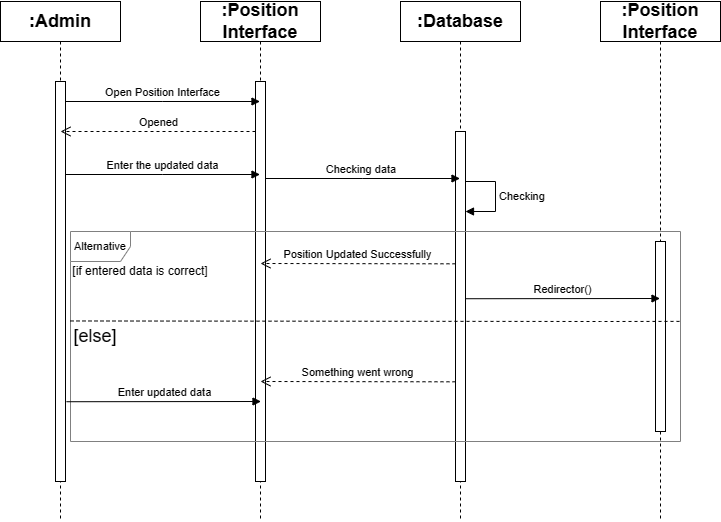


Figure 2.6.1.5.: Position Updated.

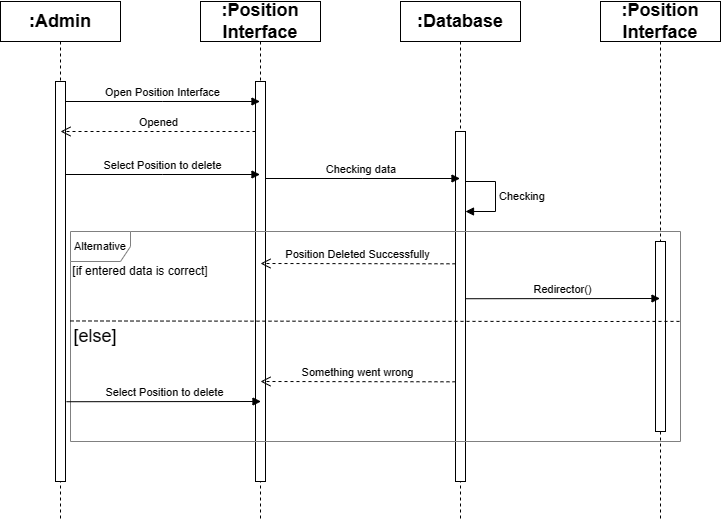


Figure 2.6.1.6.: Position Deleted.

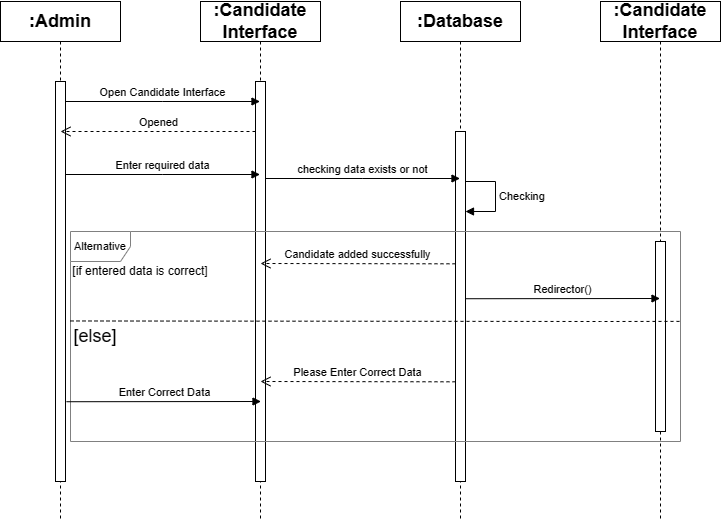


Figure 2.6.1.7.: Candidate Added.

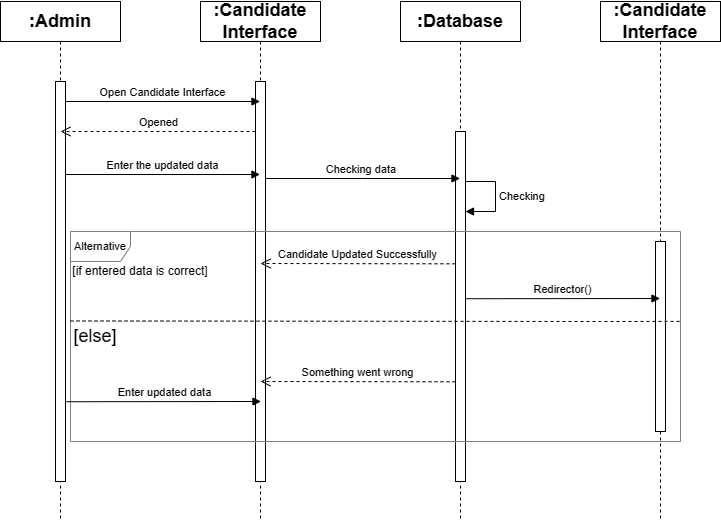


Figure 2.6.1.8.: Candidate Updated.

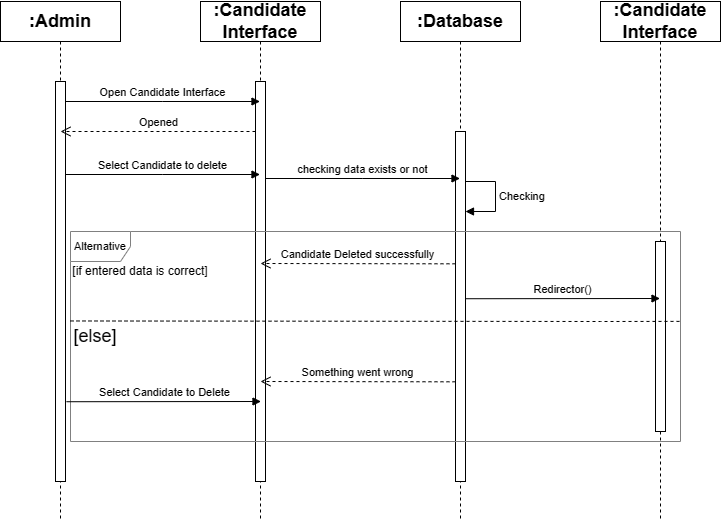


Figure 2.6.1.9.: Candidate Deleted.

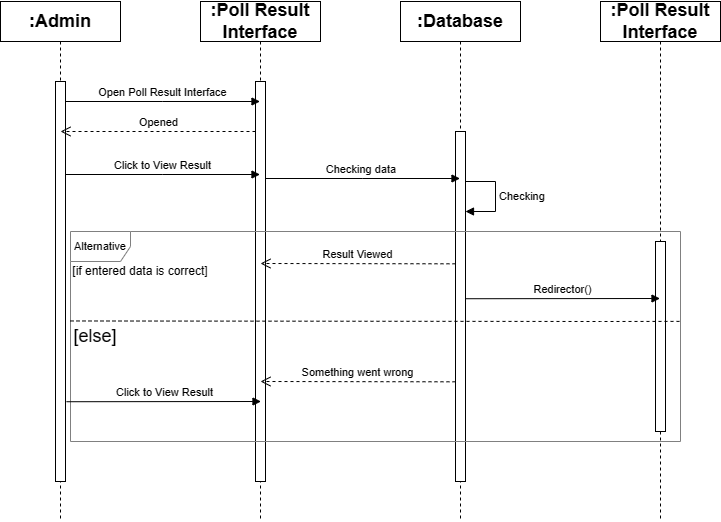


Figure 2.6.1.10.: Voting Result Viewed.

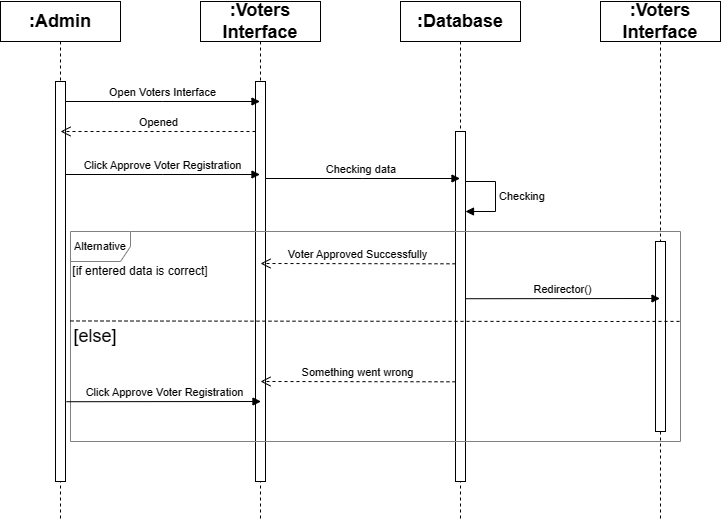


Figure 2.6.1.11.: Voter Registration Approve.

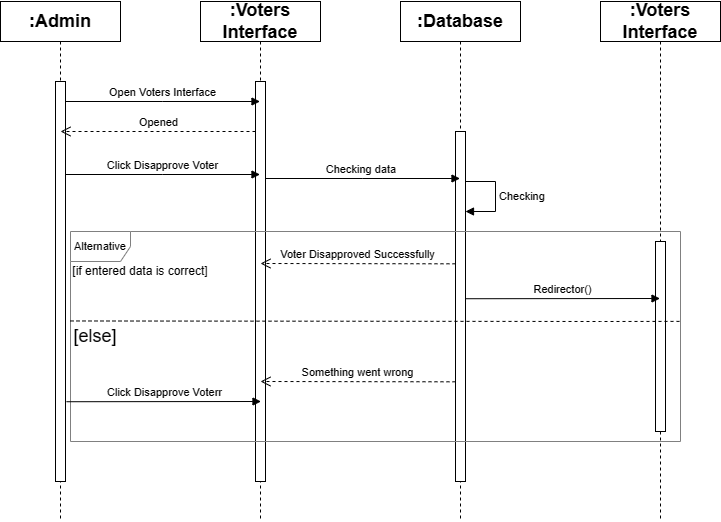


Figure 2.6.1.12.: Voter Registration Disapproved

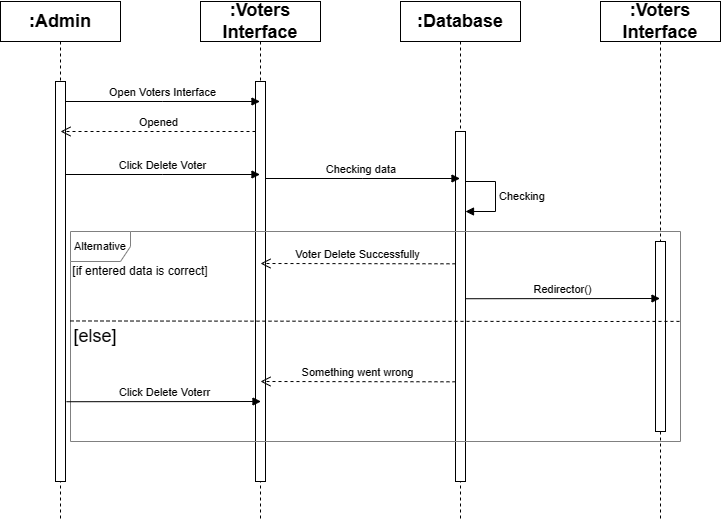


Figure 2.6.1.13.: Voter Registration Deleted.

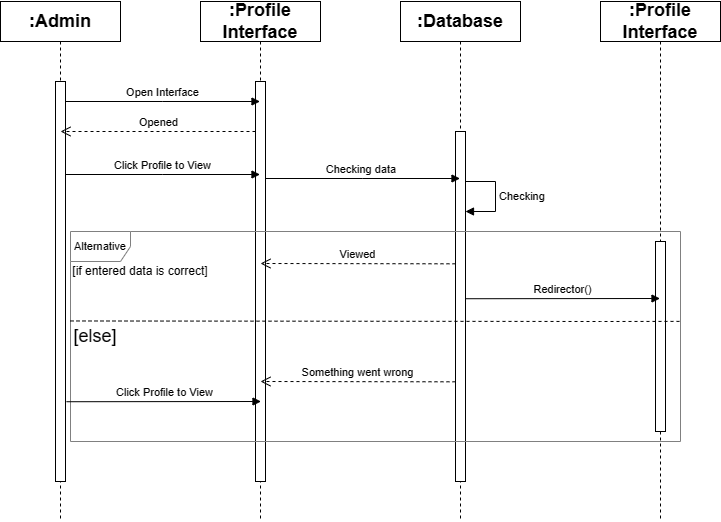


Figure 2.6.1.14.: Admin Profile Viewed.

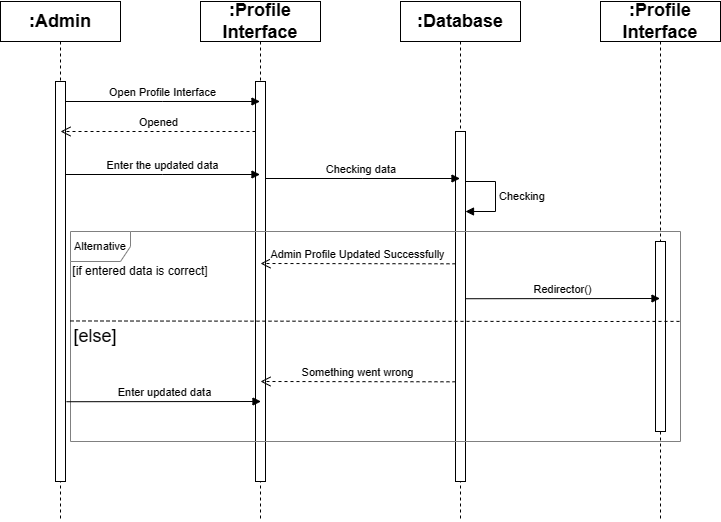


Figure 2.6.1.15.: Admin Profile Updated.

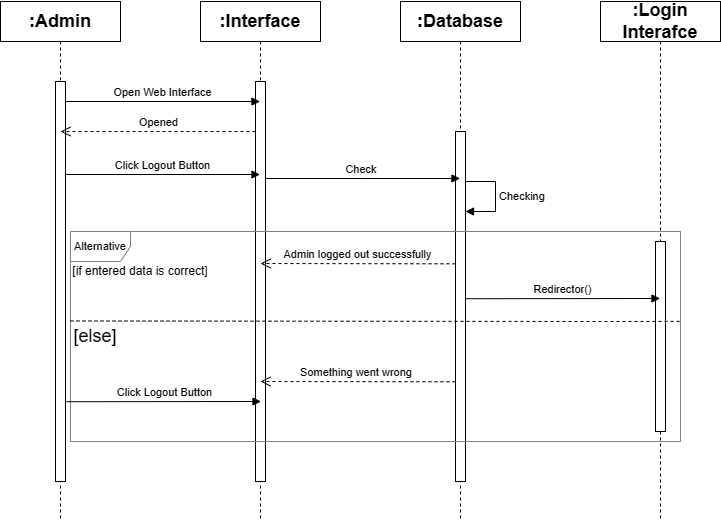


Figure 2.6.1.16.: Admin Logout.

### 2.6.2. Voter Module

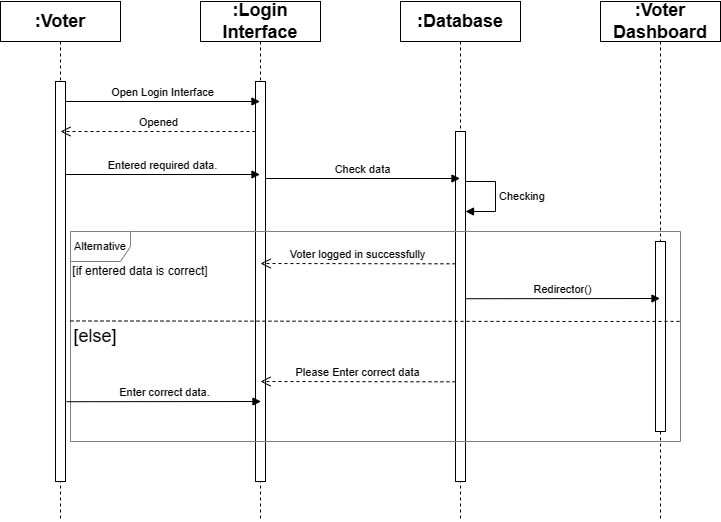


Figure 2.6.2.1: Voter Login.

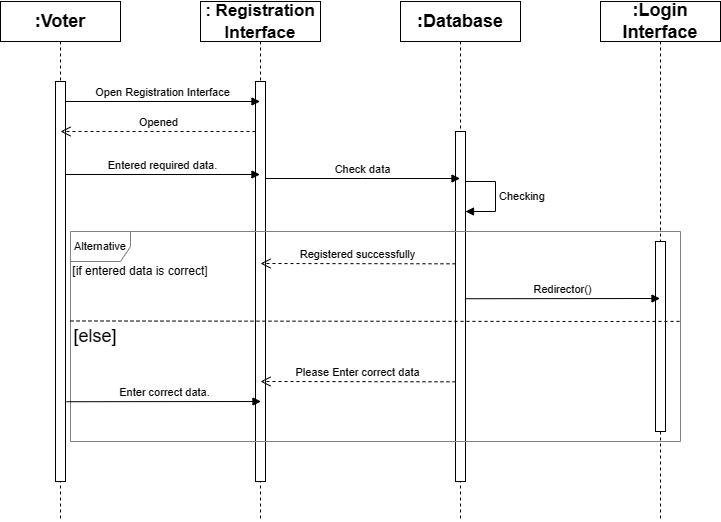


Figure 2.6.2.2: Voter Registration.

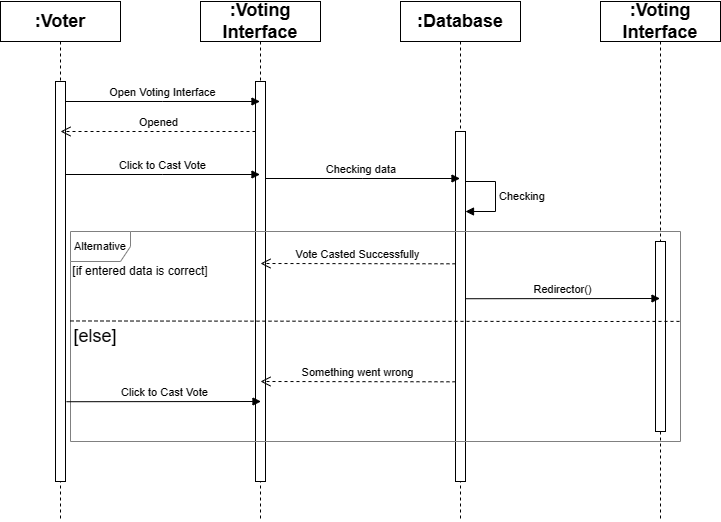


Figure 2.6.2.3: Vote Cast.

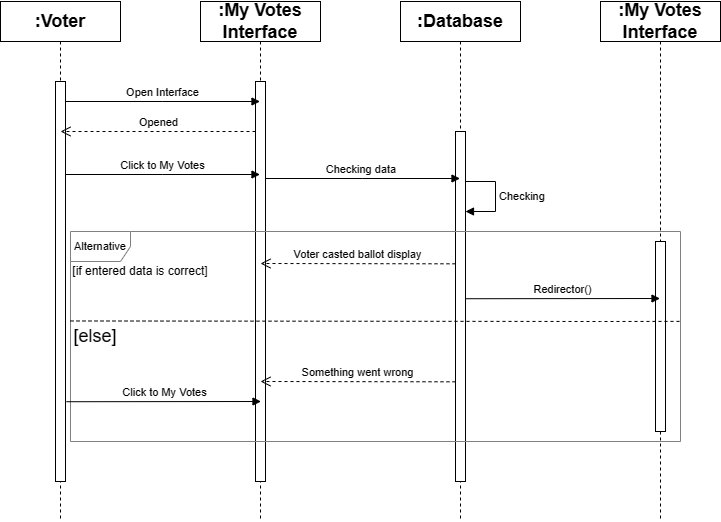


Figure 2.6.2.4: Voter’s Ballots View.

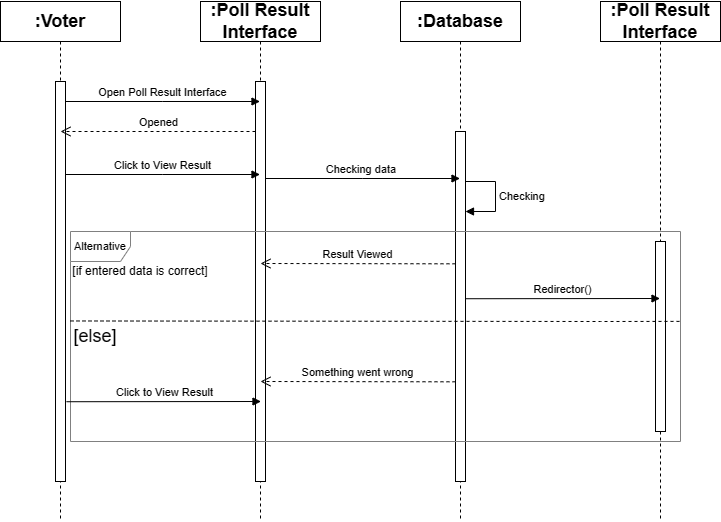


Figure 2.6.2.5: Voter View Poll Result.

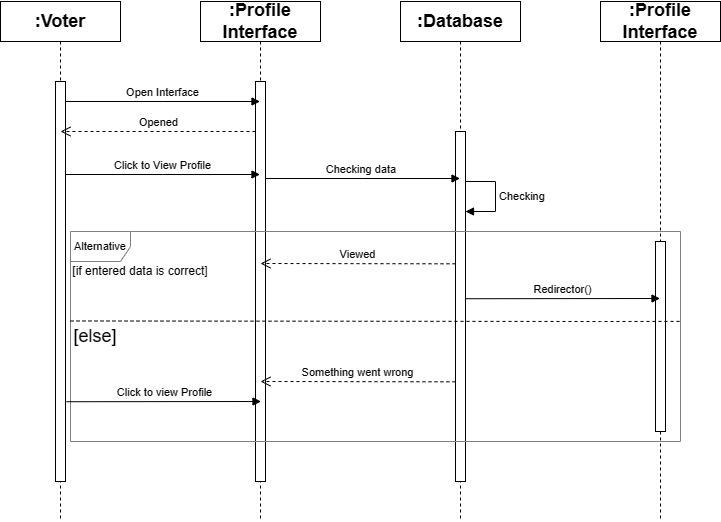


Figure 2.6.2.6: Voter Profile Viewed.

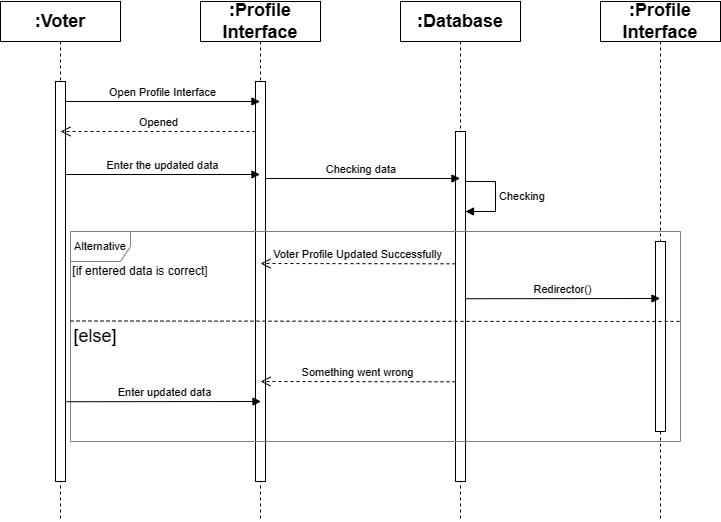


Figure 2.6.2.7: Voter Profile Updated.

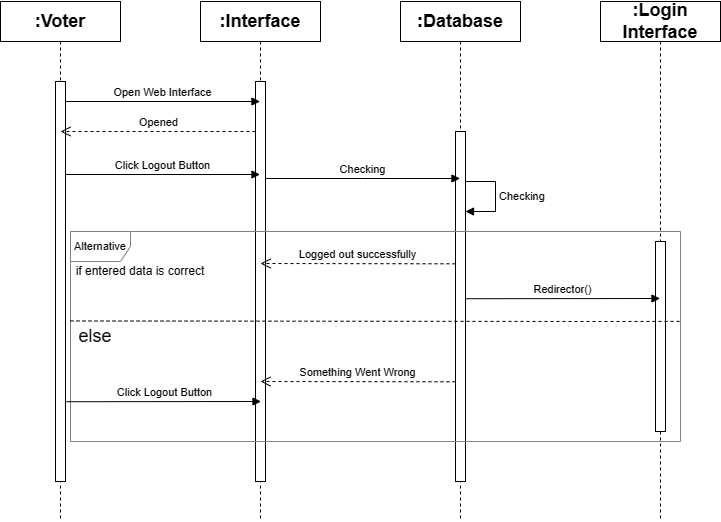


Figure 2.6.2.8: Voter Logged Out.

## 2.7. Object Model/Logical Model: Class Diagram

The Class Diagram plays a pivotal role in software development. It acts as a bridge between the conceptual object model and the concrete implementation of the application. By defining classes, attributes, methods, and relationships, Class Diagrams offer a visual representation of the system, facilitating effective communication and efficient design. Here’s the Class diagram of the Web-Based Online Voting System.

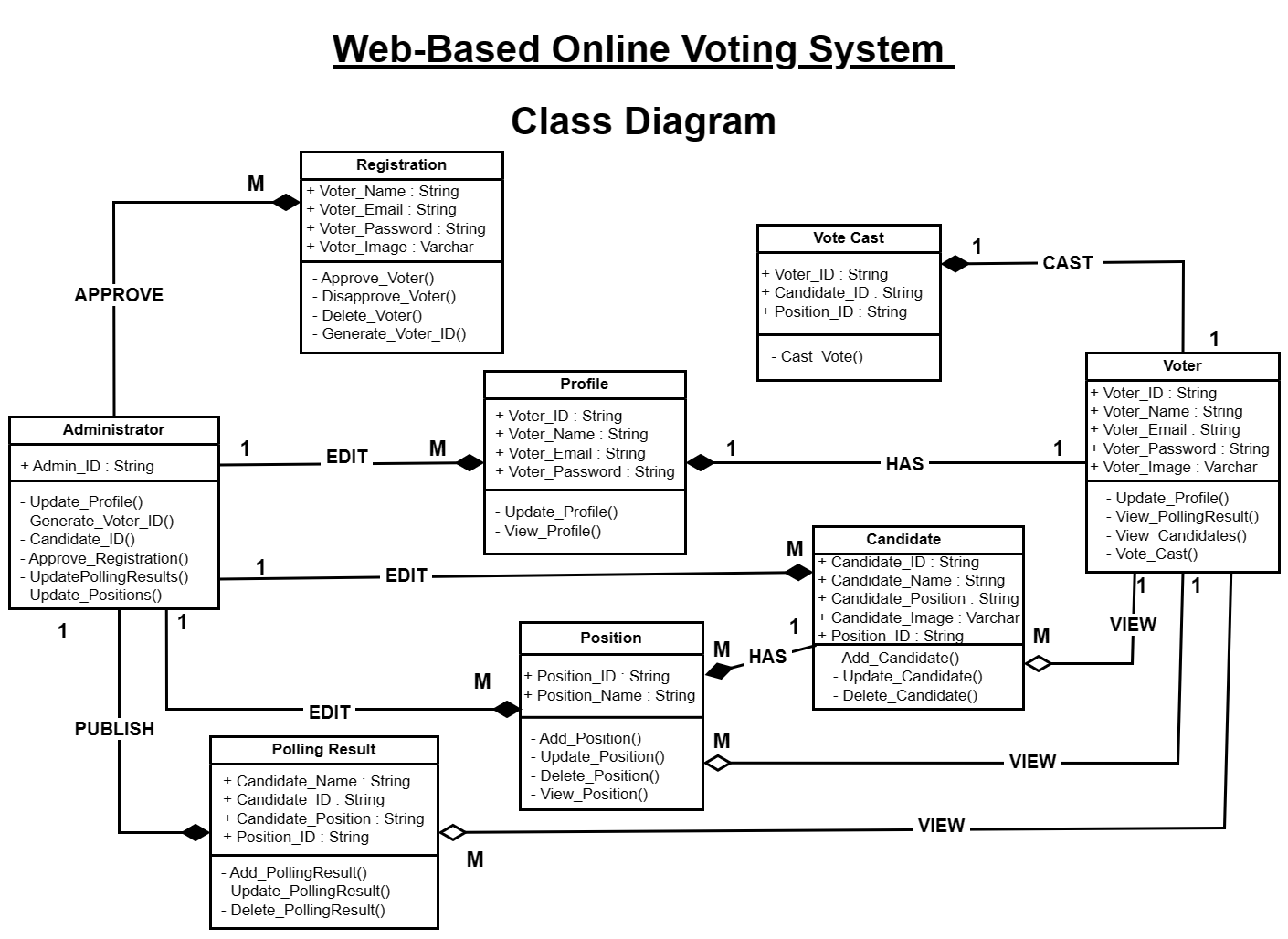


Figure 2.7.: Class Diagram.

## 2.8. Database Model (Database Diagram)

A database model is a logical representation of a database's structure, relationships, and rules. It defines how data is organized, stored, and accessed within the database system. A database diagram is a visual representation of a database model. It uses symbols and notation to depict entities, attributes, relationships, and the overall structure of the database. The purpose of a database diagram is to provide a visual aid that helps stakeholders understand the logical structure of the database, making it easier to design, develop, and maintain the database. Here is the database diagram of a W*eb-Based online voting system*.

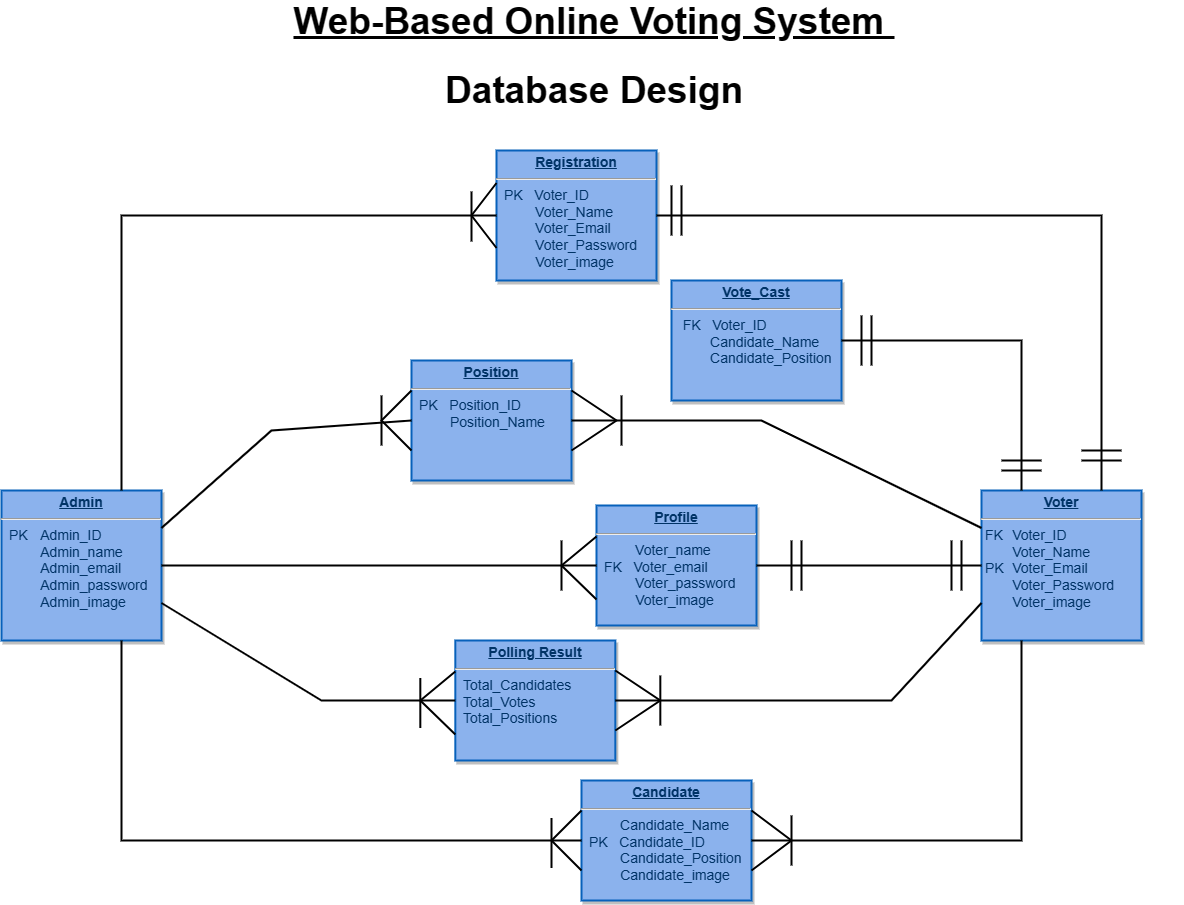


Figure 2.8.: Database Diagram.

## 2.9. Graphical User Interfaces

A Graphical User Interface (GUI) is a visual representation of an operating system or software application that allows users to interact with computers and devices using intuitive icons, menus, and graphical elements.

GUIs offer enhanced usability, intuitive navigation, reduced learning curves, and make multitasking more accessible for users. GUIs include icons, menus, toolbars, windows, and pointing devices like mice and touchscreens. GUIs are more user-friendly and accessible, while CLI relies on text commands, which can be intimidating for non-technical users.

Following are the GUIs of our voting system.

### 2.9.1. Admin Module GUIs

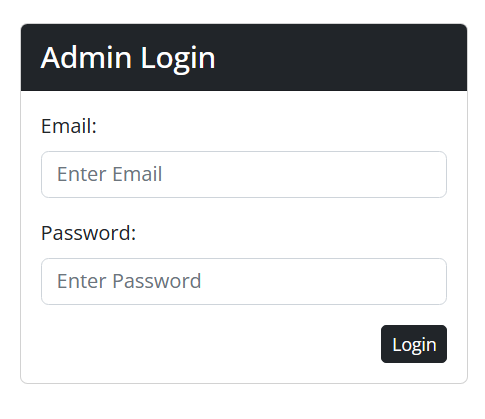


Figure 2.9.1.1.: Admin Login.

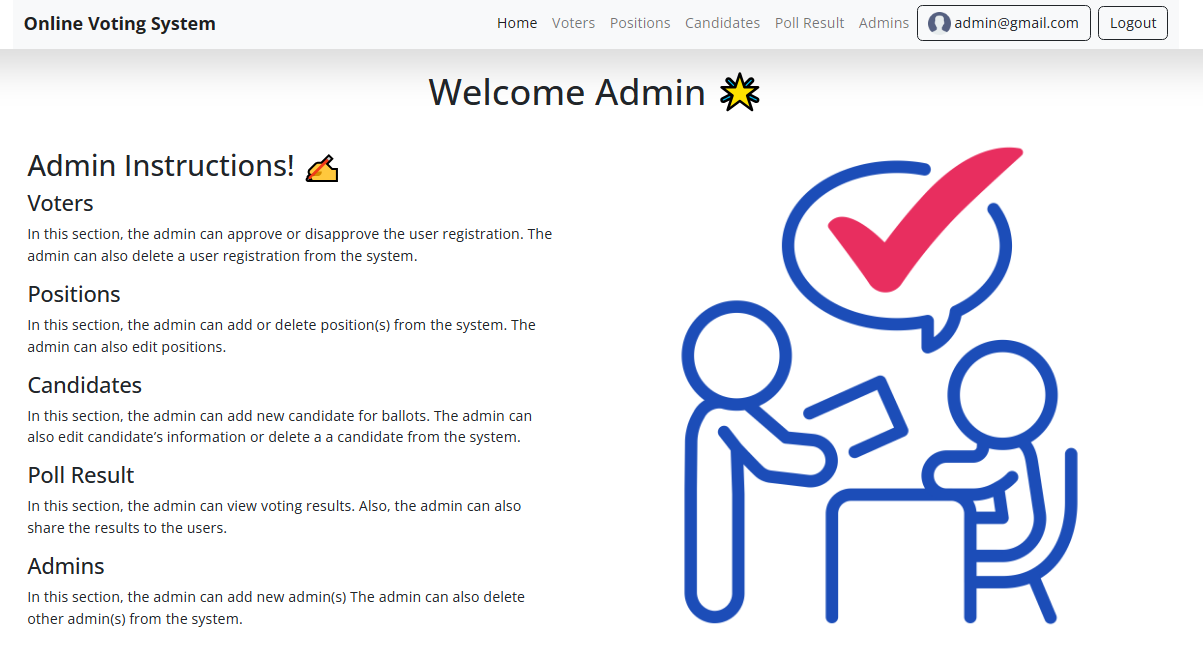


Figure 2.9.1.2.: Admin Welcome Page.

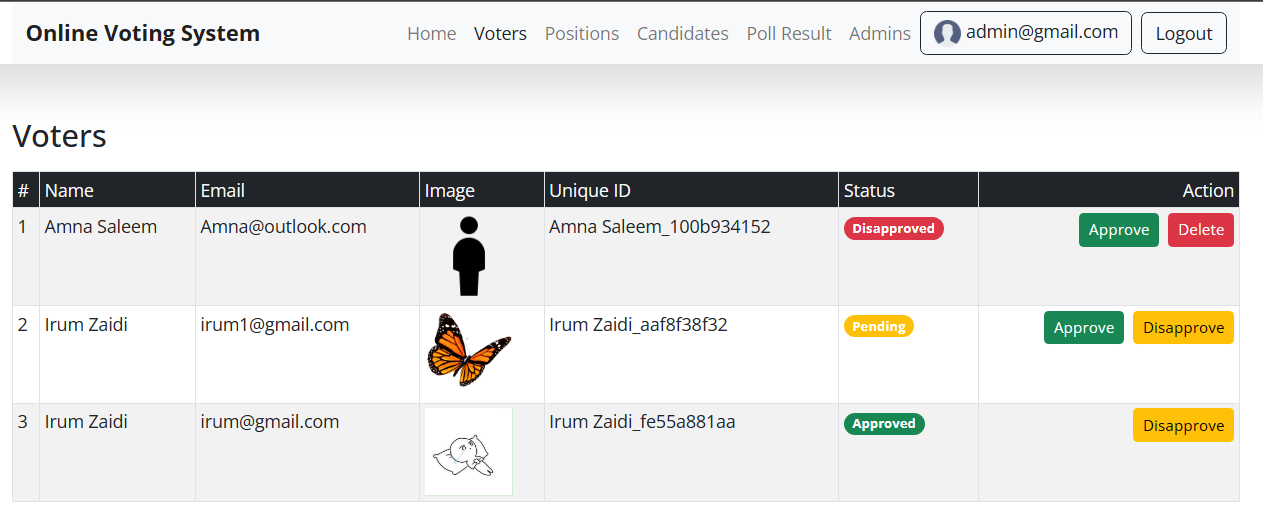


Figure 2.9.1.3.: Voters Registration.

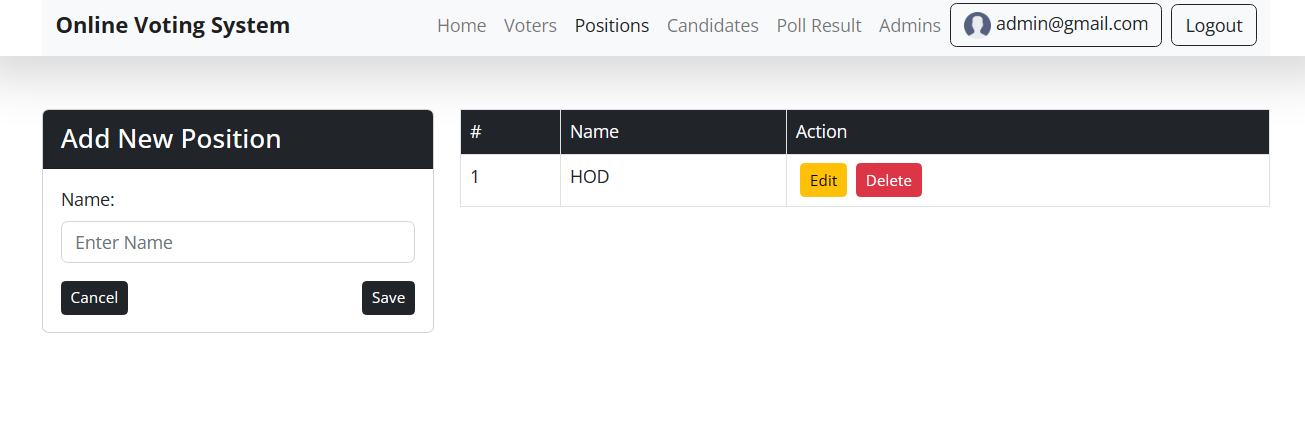


Figure 2.9.1.4.: Voting Position.

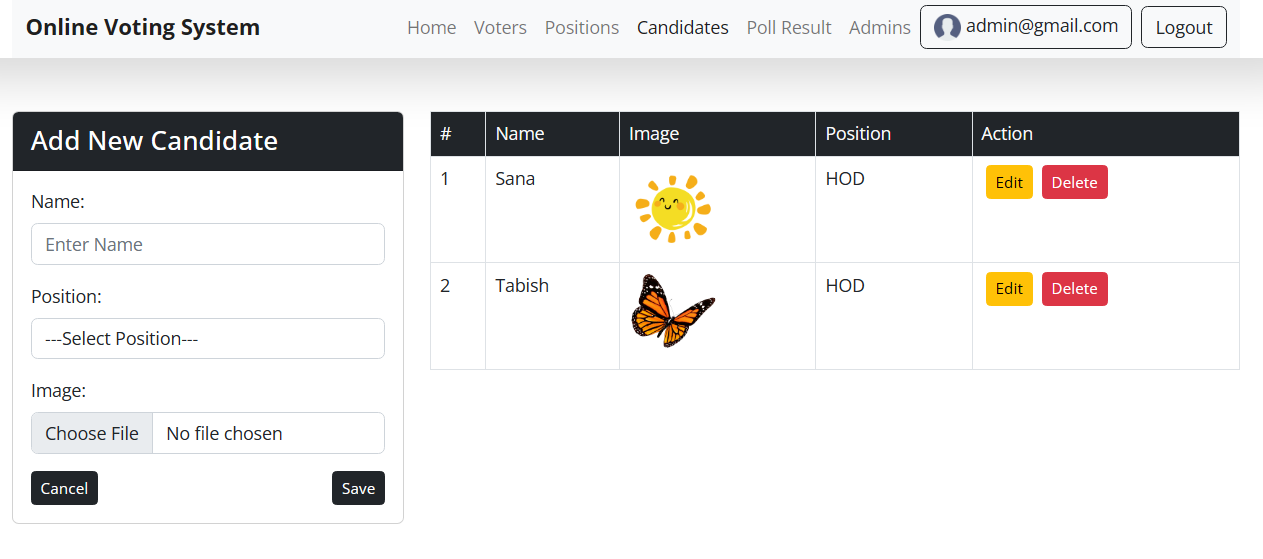


Figure 2.9.1.5.: Candidates.

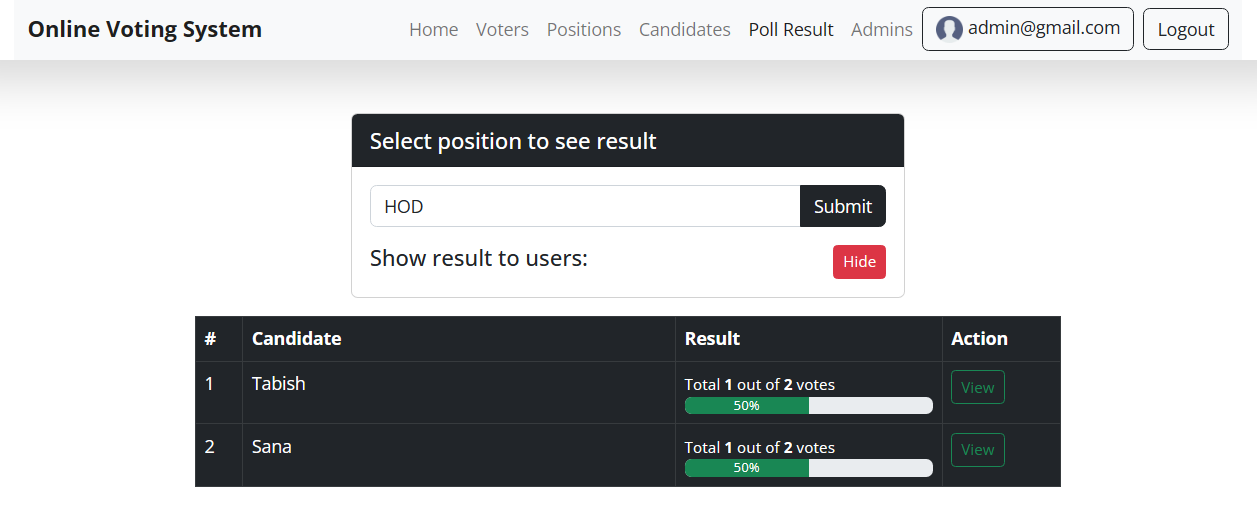


Figure 2.9.1.6.: Poll Result.

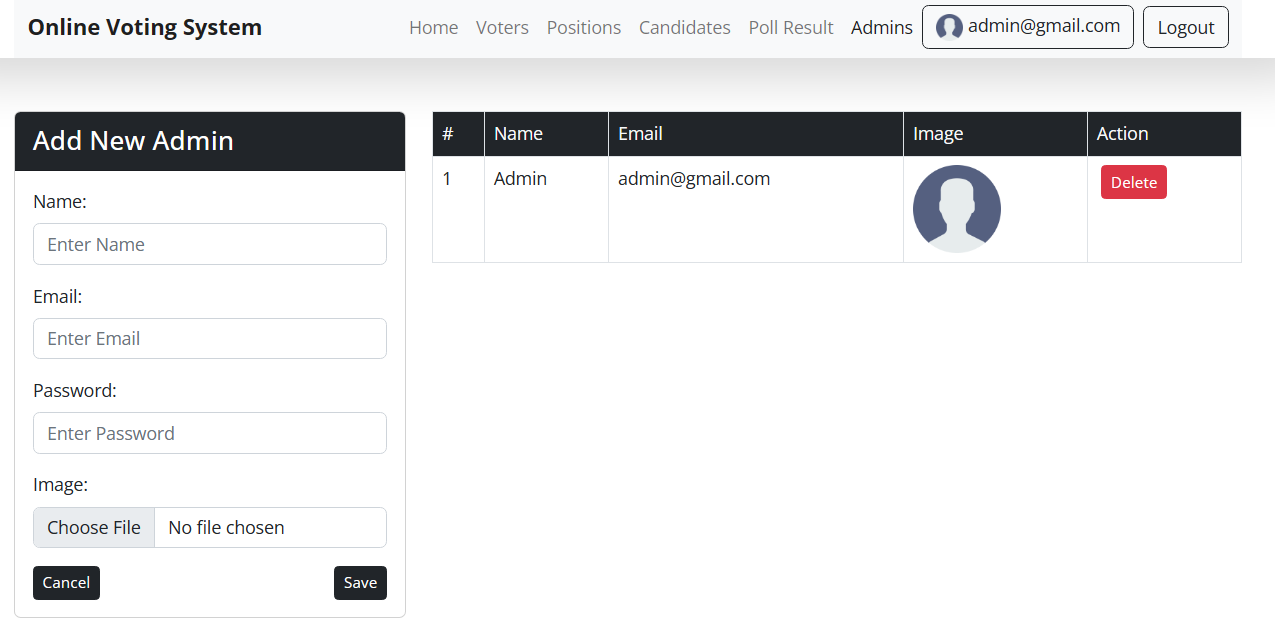


Figure 2.9.1.7.: Admins.

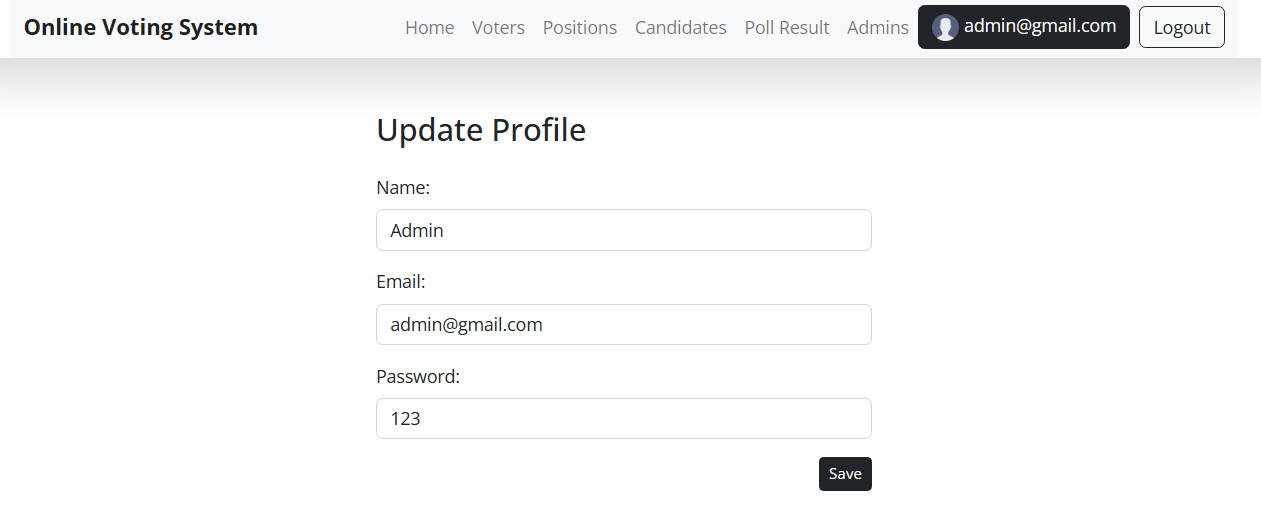


Figure 2.9.1.8.: Admin Profile.

### 2.9.2. Voter Module GUIs

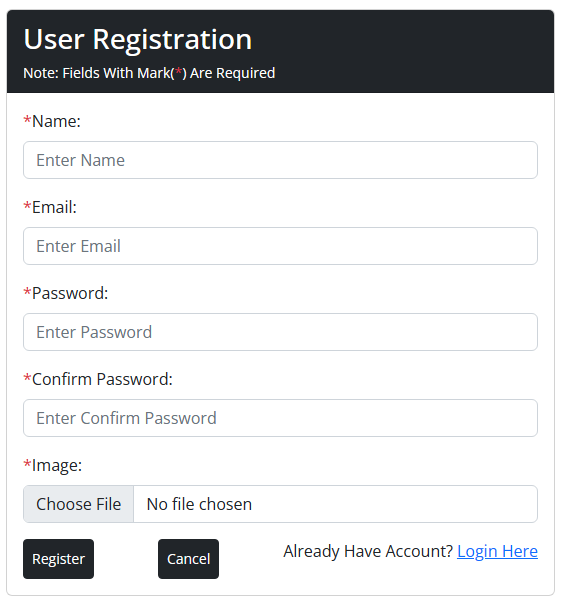


Figure 2.9.2.1.: Voter Registration.

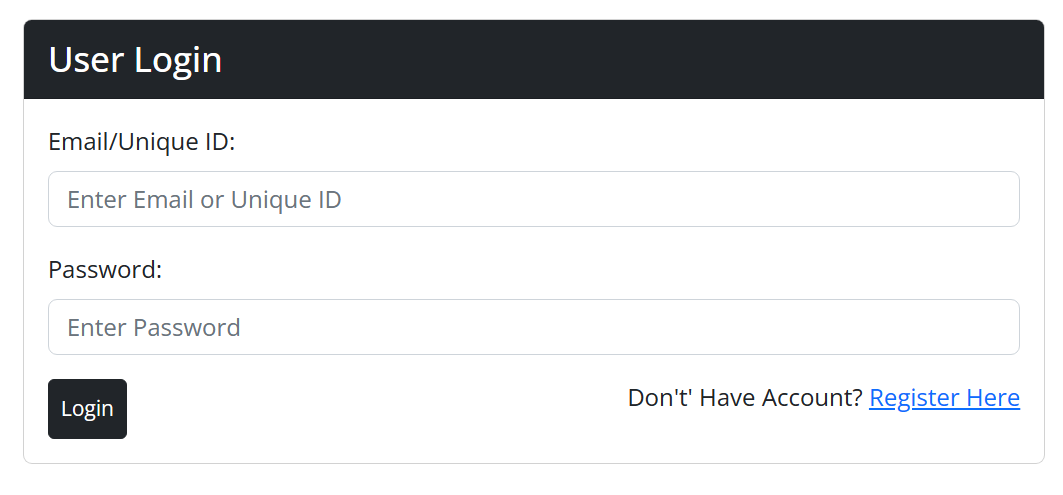


Figure 2.9.2.2.: Voter Login.



Figure 2.9.2.3.: Voter Welcome Page.

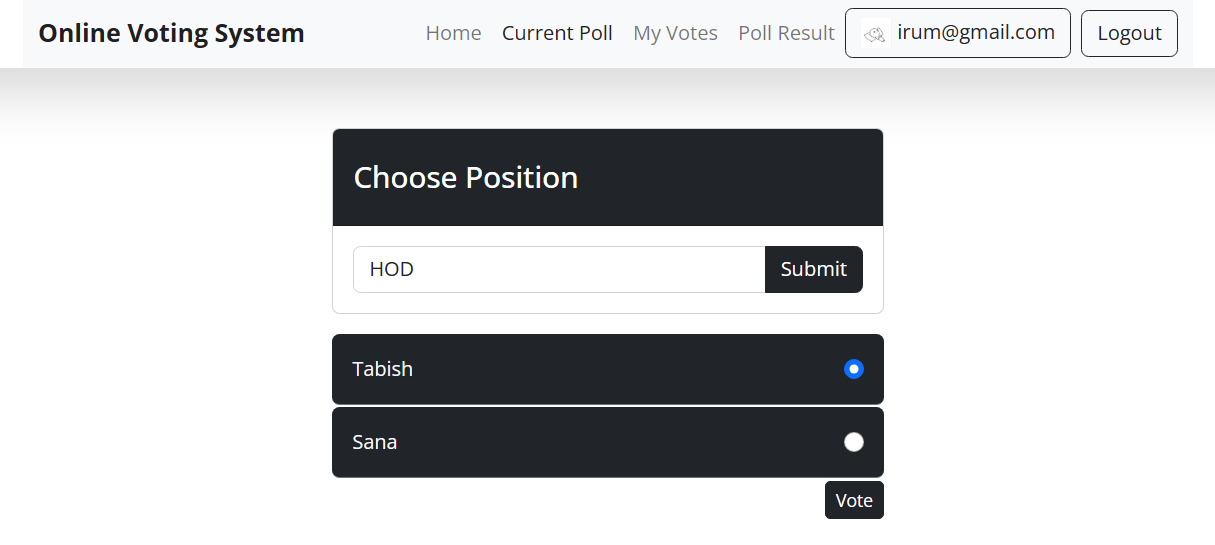


Figure 2.9.2.4.: Voting Page.

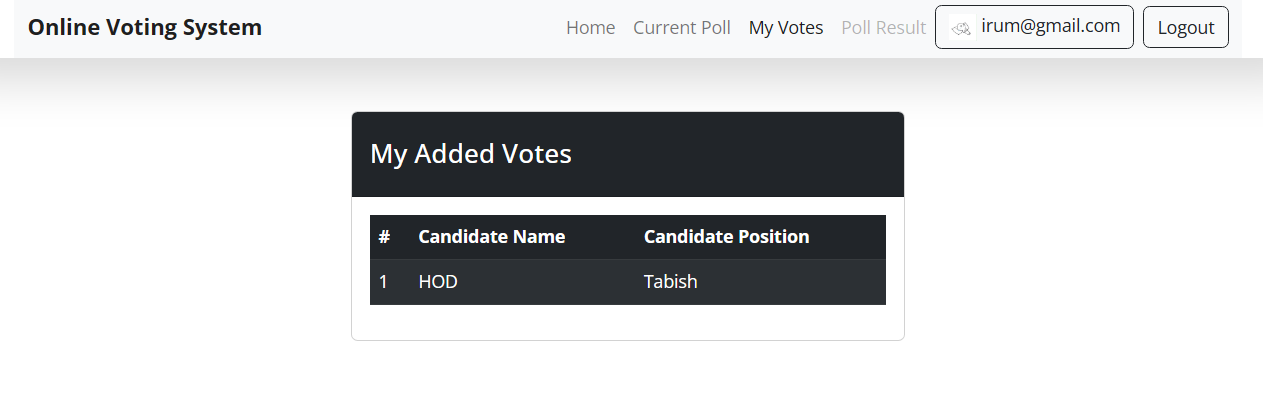


Figure 2.9.2.5.: Ballot cast by voters.

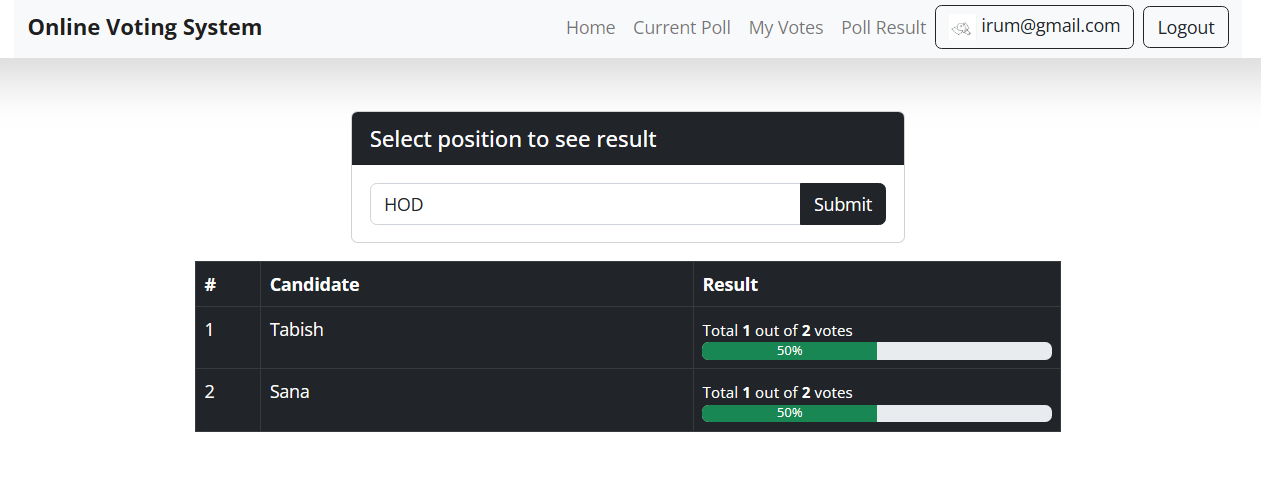


Figure 2.9.2.6.: Poll Result.

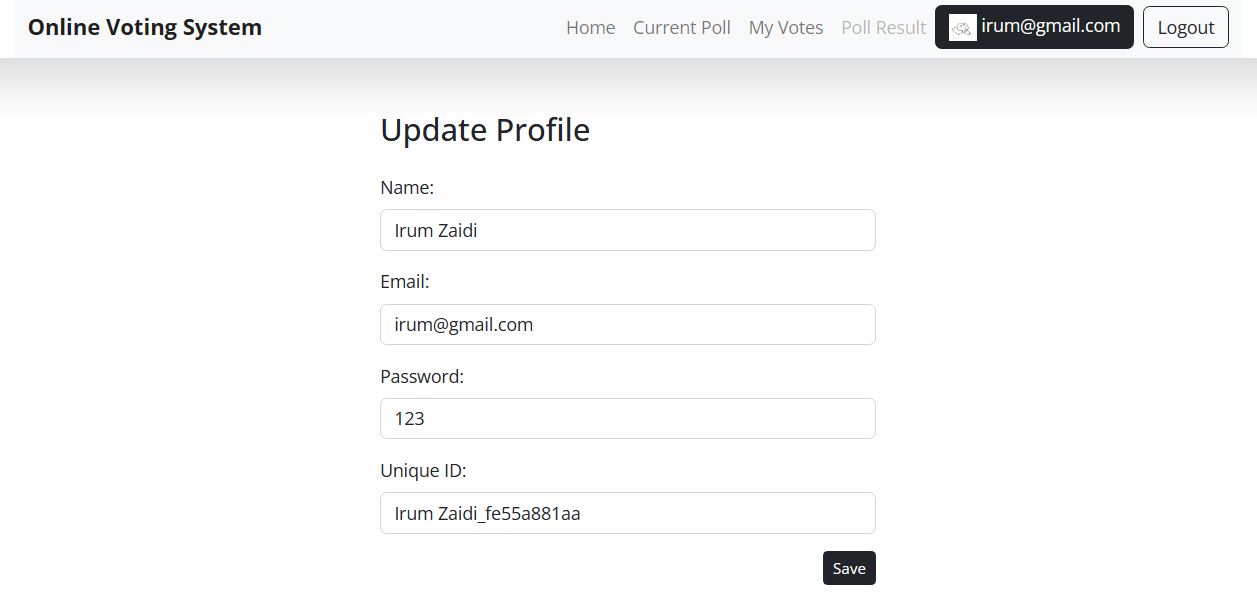


Figure 2.9.2.7.: Voter Profile.

**CHAPTER 3**

Development

## 3.1. Development plan (Architecture Diagram)

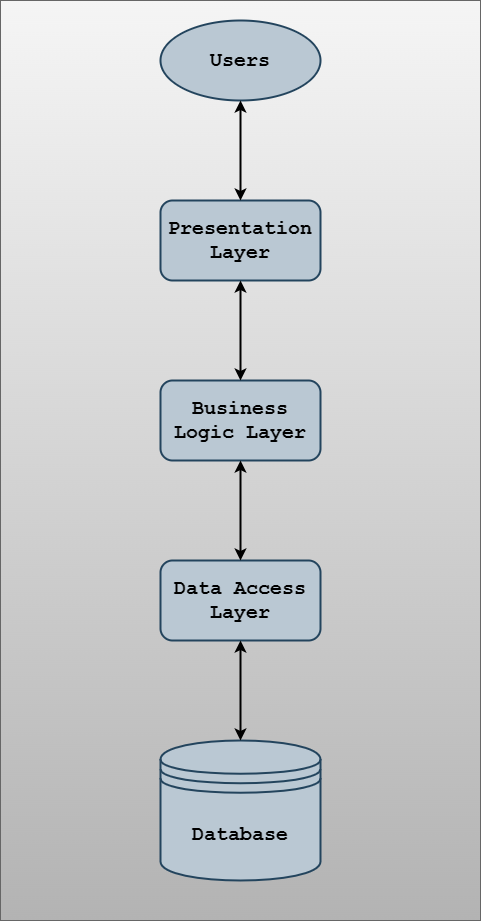


Figure 3.1: Architecture Design.

**REFERENCES**

* <https://www.youtube.com/>
* <https://stackoverflow.com/>
* <https://www.w3schools.com.com/>
* <https://www.apachefriends.org/>
* <https://www.udemy.com/>

**APPENDIX**

| Section#1 | GATHERING & ANALYZING INFO. |
| --- | --- |
| Section#2 | DESIGNING PROJECT. |
| Section#3 | DEVELOPMENT. |

* End of Document.