4.2.1 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.

Sequence Diagram Notations –

i. **Actors –** An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram. We use actors to depict various roles including human users and other external subjects. We represent an actor in a UML diagram using a stick person notation. We can have multiple actors in a sequence diagram.

ii. **Lifelines –** A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically, each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram.

iii. **Messages –** Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram.

Messages can be broadly classified into the following categories:

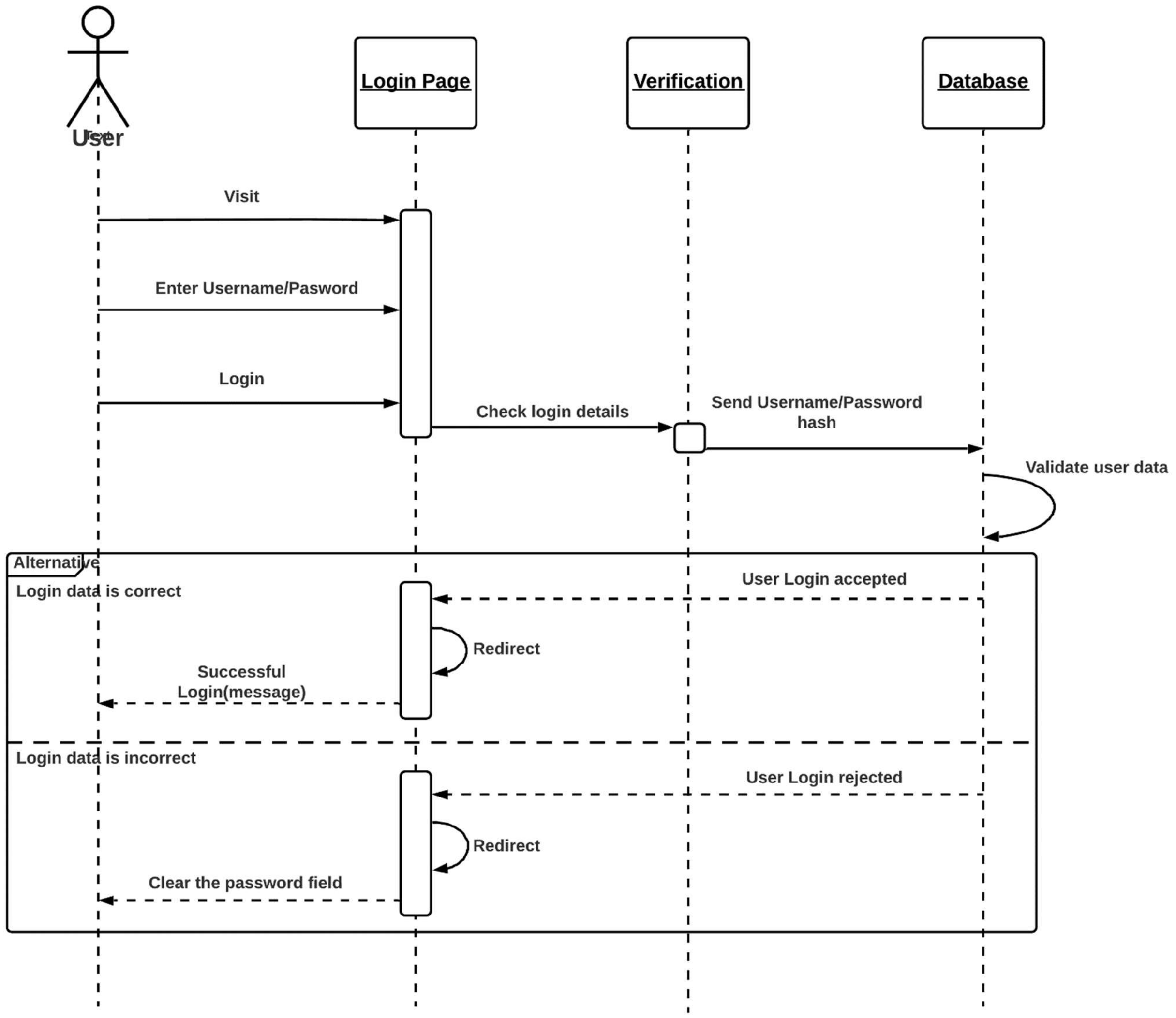
- Synchronous messages
- Asynchronous Messages
- Create message
- Delete Message
- Self-Message
- Reply Message
- Found Message
- Lost Message

iv. **Guards –** To model conditions we use guards in UML. They are used when we need to restrict the flow of messages on the pretext of a condition being met. Guards play an important role in letting software developers know the constraints attached to a system or a particular process.

Uses of sequence diagrams –

- Used to model and visualize the logic behind a sophisticated function, operation or procedure.
- They are also used to show details of UML use case diagrams.
- Used to understand the detailed functionality of current or future systems.
- Visualize how messages and tasks move between objects or components in a system.

Fig 1: Sequence diagram for Online Voting System(OVS) Login page



4.2.2 State Chart Diagram

State chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of State chart diagram is to model lifetime of an object from creation to termination.

Fig 1: Super admin State chart diagram for Online Voting System(OVS)

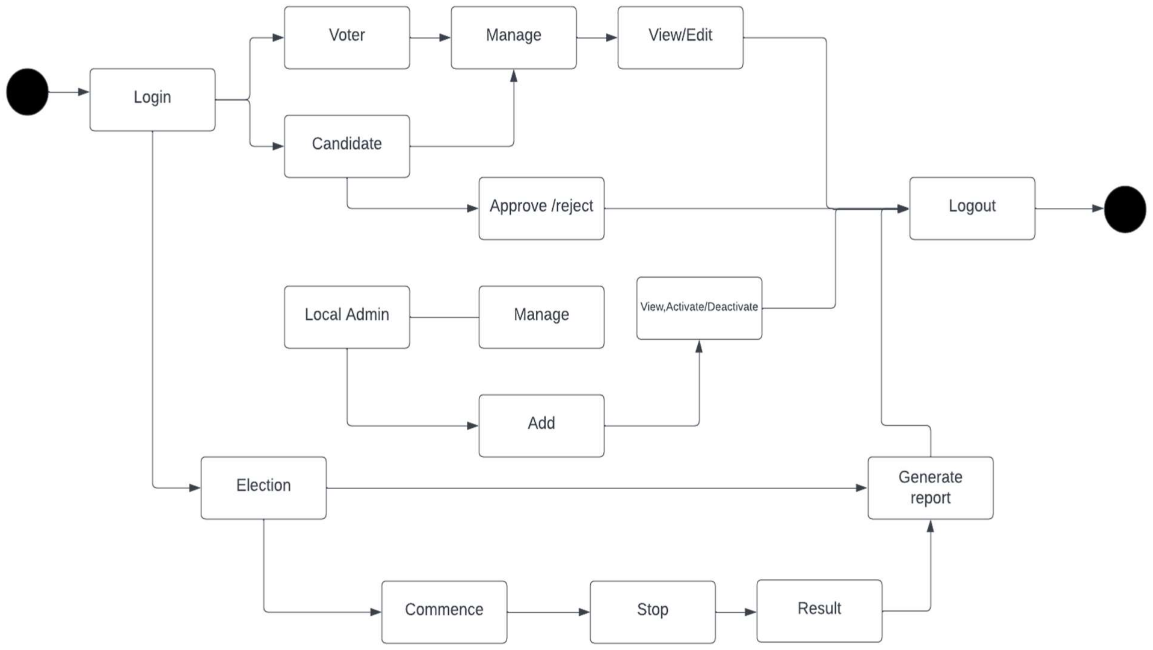


Fig 2: Local admin State chart diagram for Online Voting System(OVS)

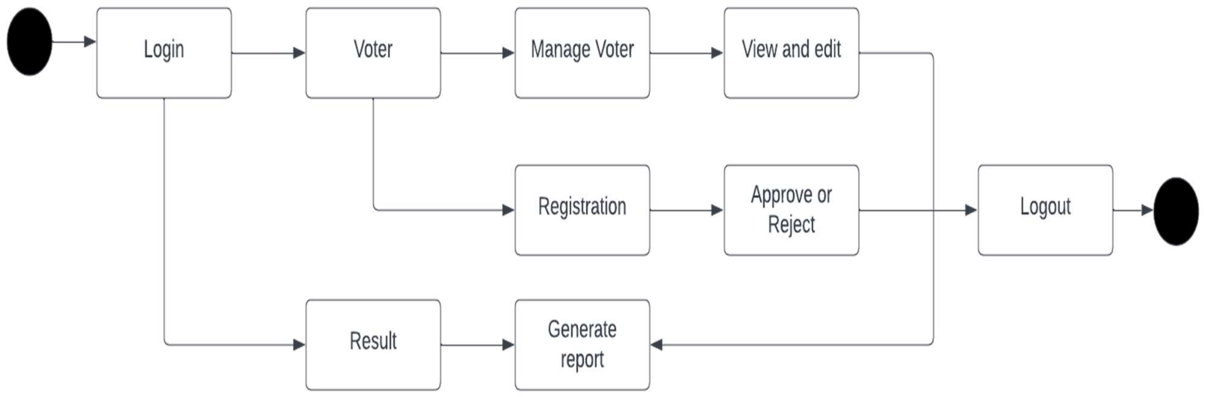


Fig 3 : Candidate State chart diagram for Online Voting System(OVS)

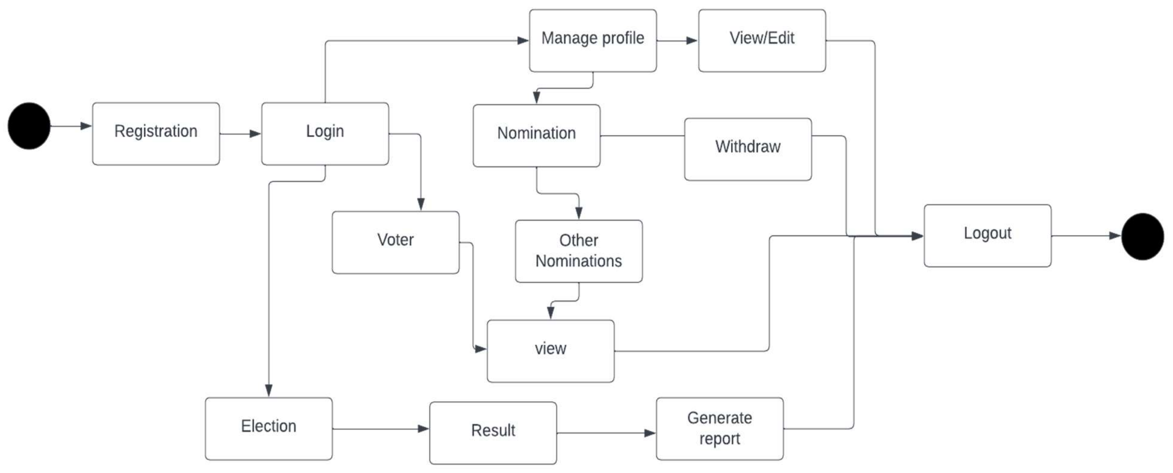
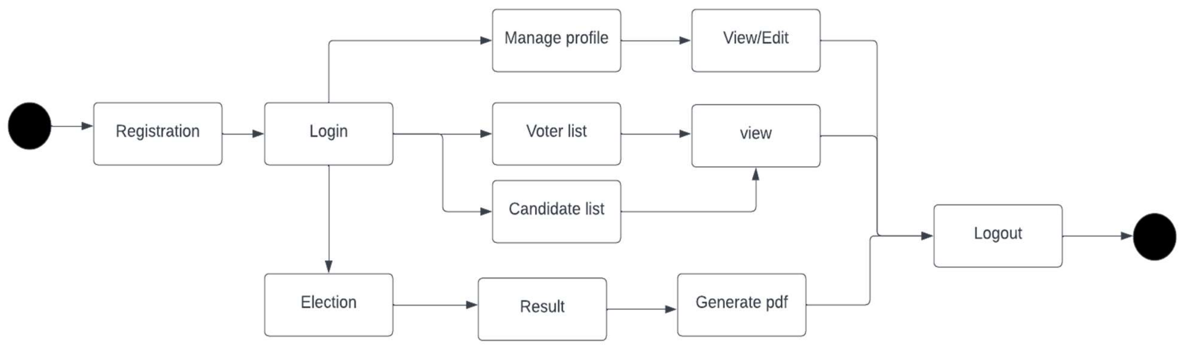


Fig 4 : Voter State chart diagram for Online Voting System(OVS)



4.2.3 Activity Diagram

We use Activity Diagrams to illustrate the flow of control in a system and 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.

Fig 1: Admin Activity diagram for Online Voting System(OVS)

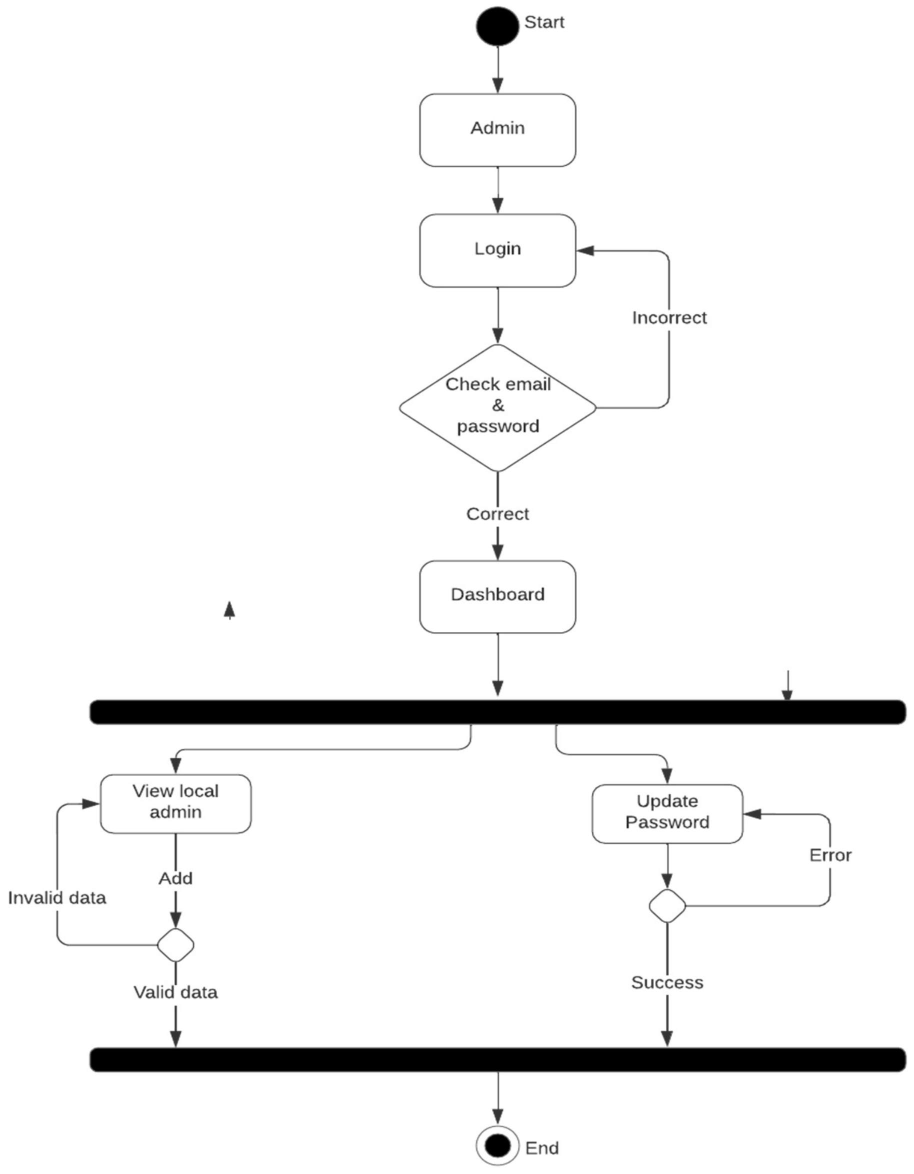


Fig 2: Subadmin Activity diagram for Online Voting System(OVS)

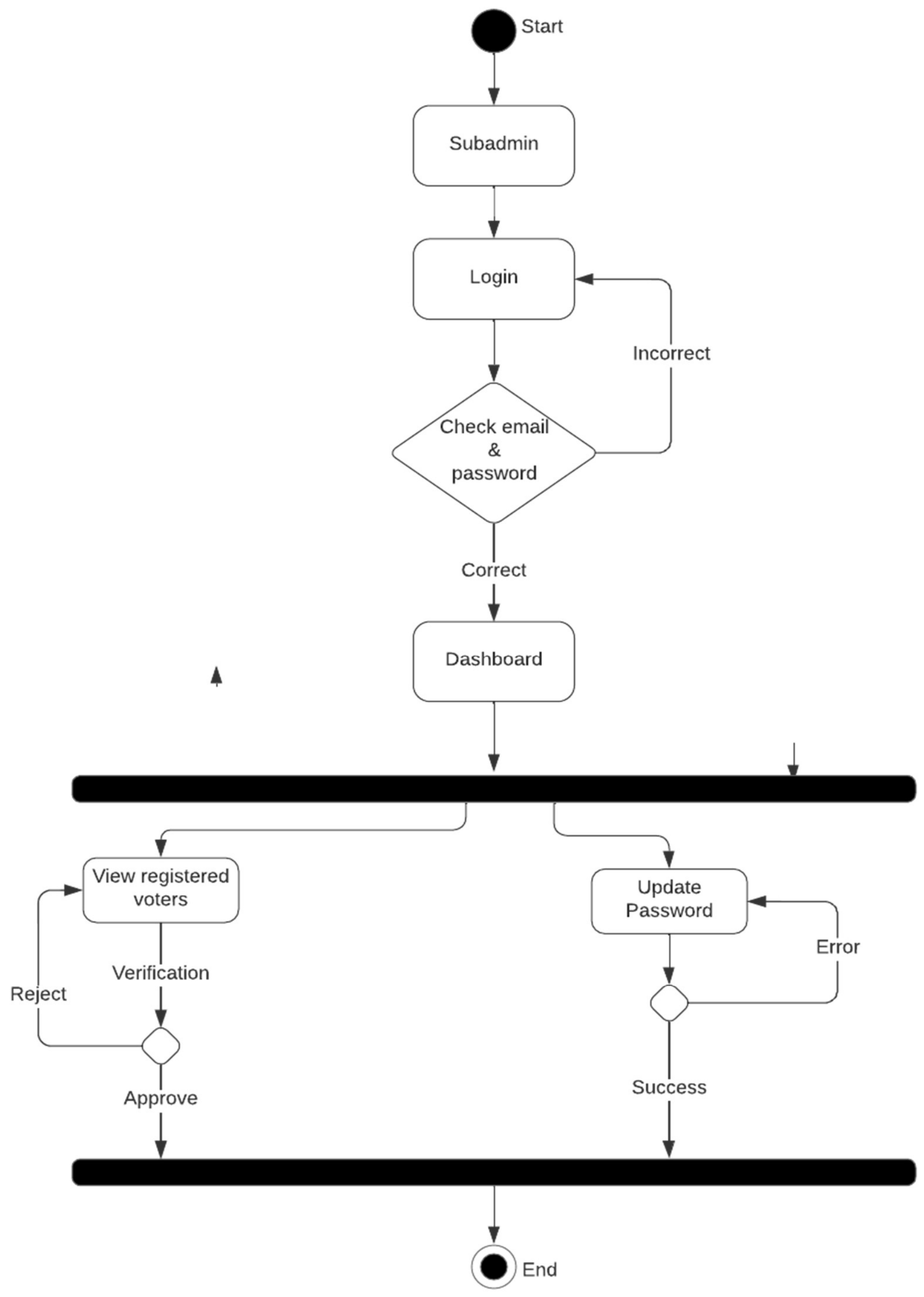


Fig 2: Voter Activity diagram for Online Voting System(OVS)

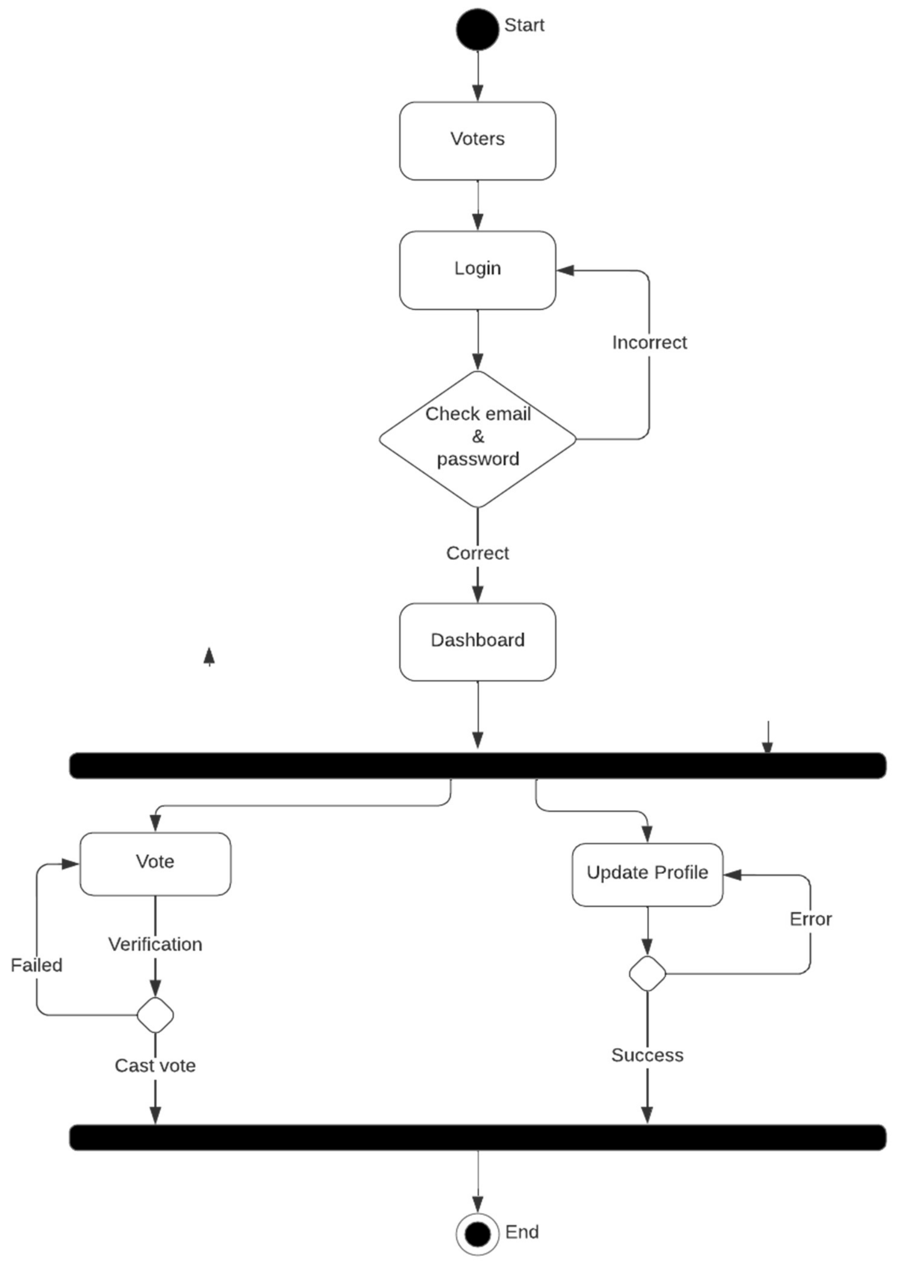
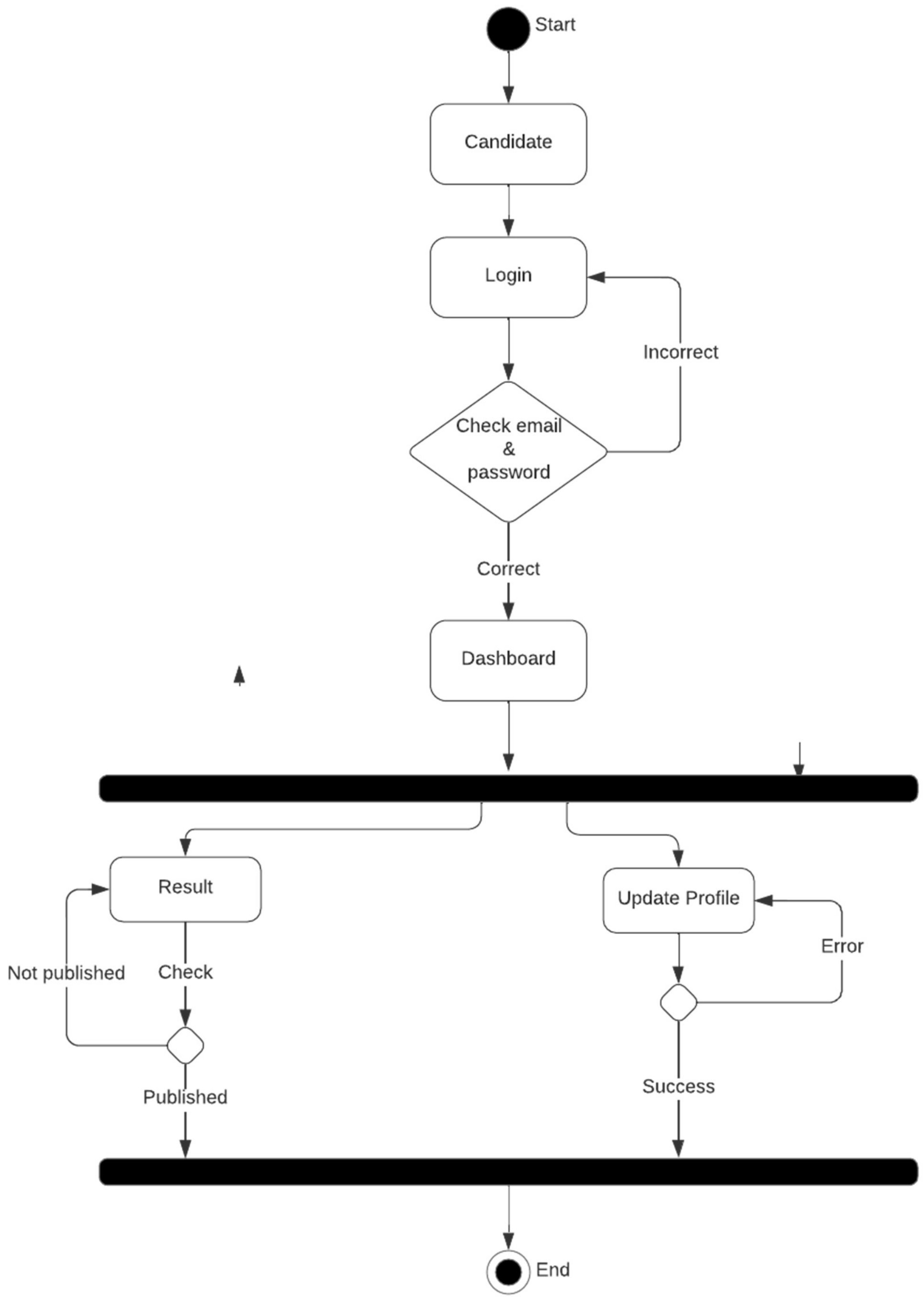


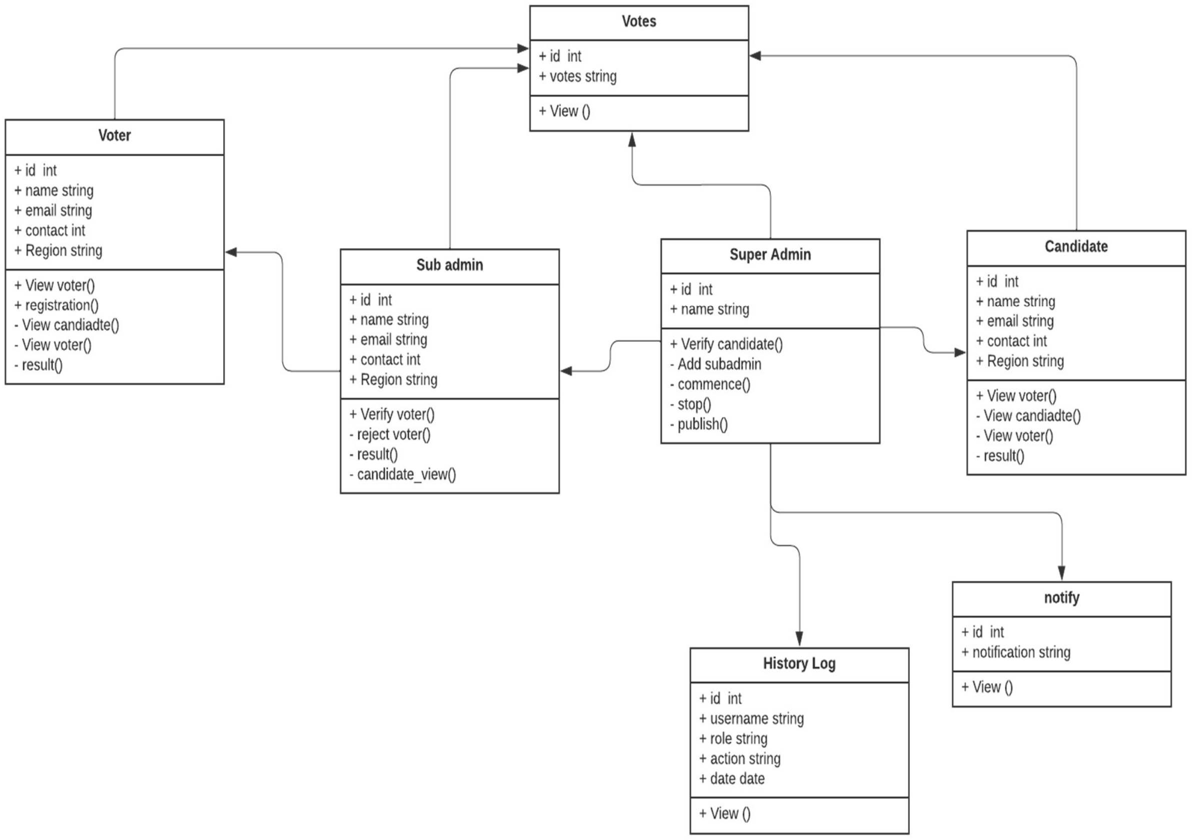
Fig 3: Candidate Activity diagram for Online Voting System(OVS)



4.2.4 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

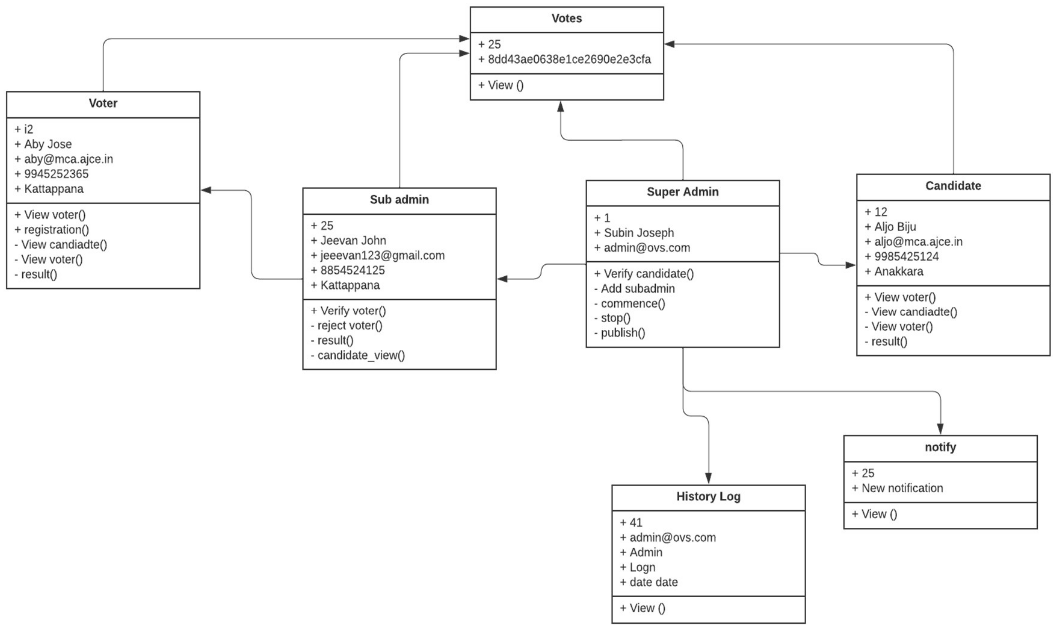


4.2.5 Object Diagram

Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams. Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment. Object diagrams are used to render a set of objects and their relationships as an instance.

The purpose of the object diagram can be summarized as

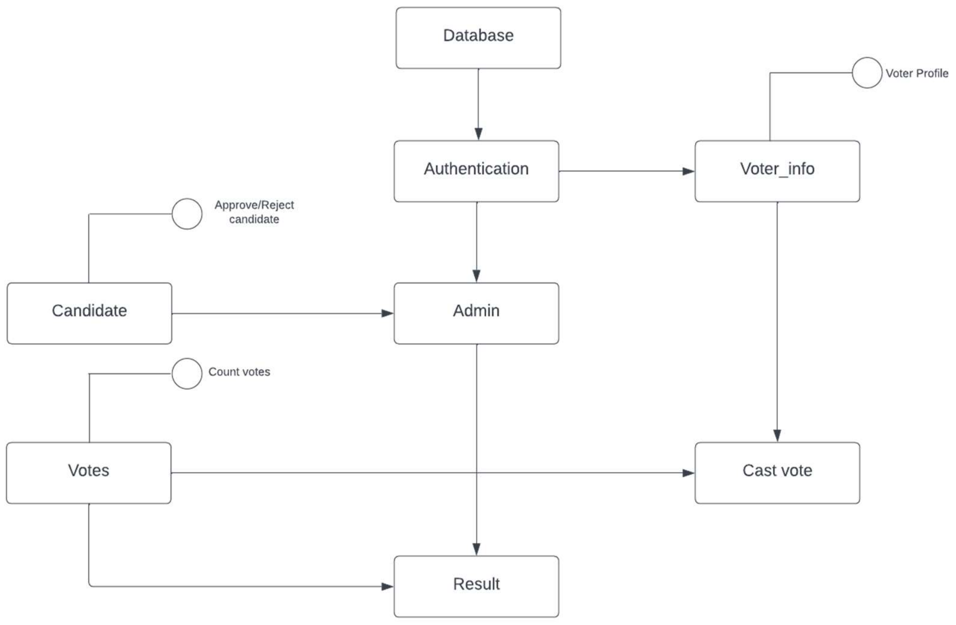
- Forward and reverse engineering.
- Object relationships of a system
- Static view of an interaction.
- Understand object behavior and their relationship from practical perspective



4.2.6 Component Diagram

UML Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.

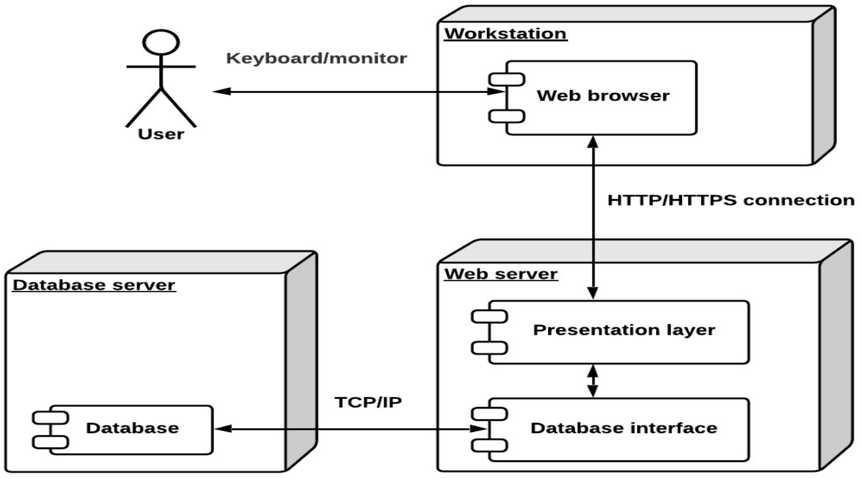
A component diagram breaks down the actual system under development into various high levels of functionality. Each component is responsible for one clear aim within the entire system and only interacts with other essential elements on a need-to-know basis.



4.2.8 Deployment Diagram

A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them.
Deployment diagrams are typically used to visualize the physical hardware and software of a system. Using it you can understand how the system will be physically deployed on the hardware.
Deployment diagrams help model the hardware topology of a system compared to other UML diagram types which mostly outline the logical components of a system.

Fig 3: Deployment diagram for Online Voting System(OVS)



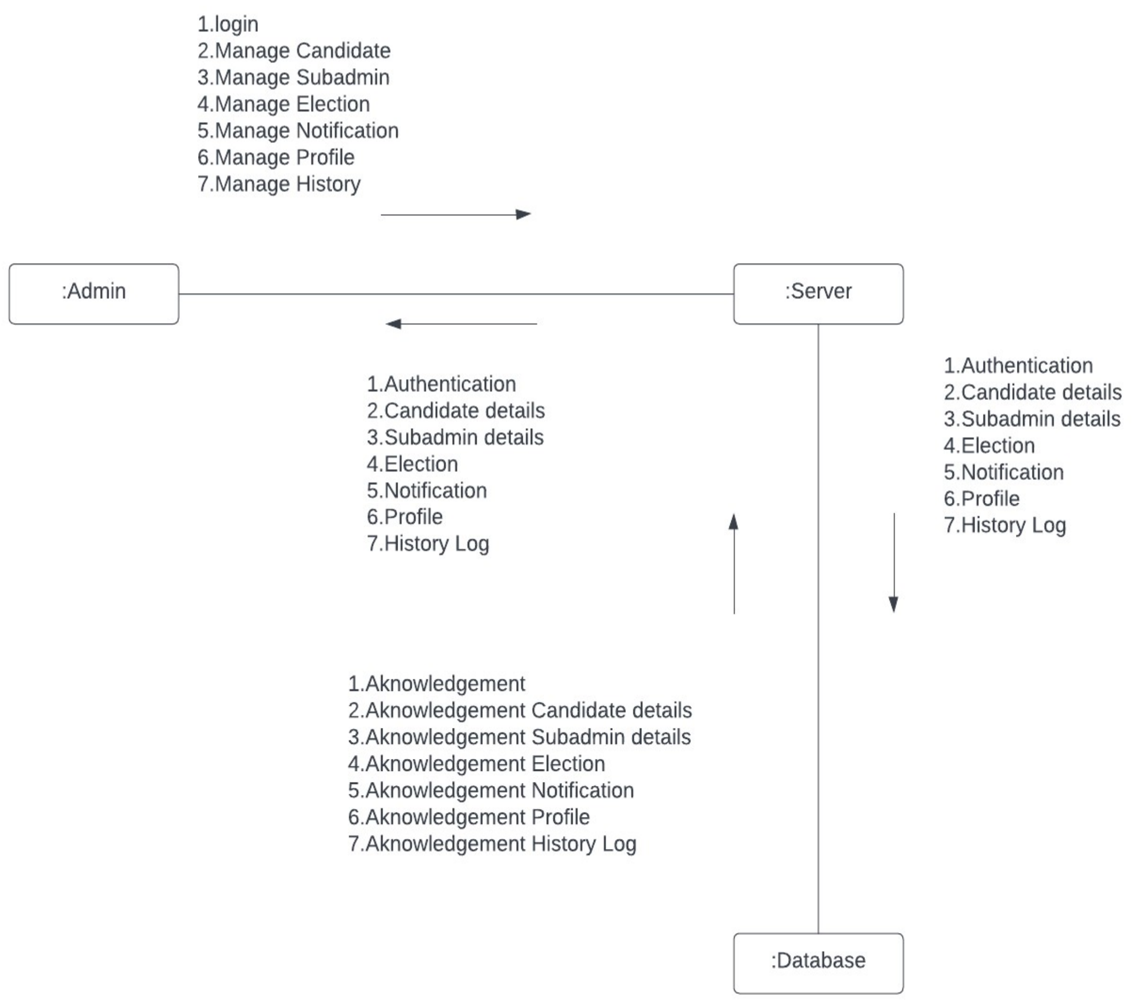
4.2.9 Collaboration Diagram

The Collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.
Following are the components of a component diagram that are enlisted below:

- Objects: The representation of an object is done by an object symbol with its name and class underlined, separated by a colon. In the collaboration diagram, objects are utilized in the following ways:

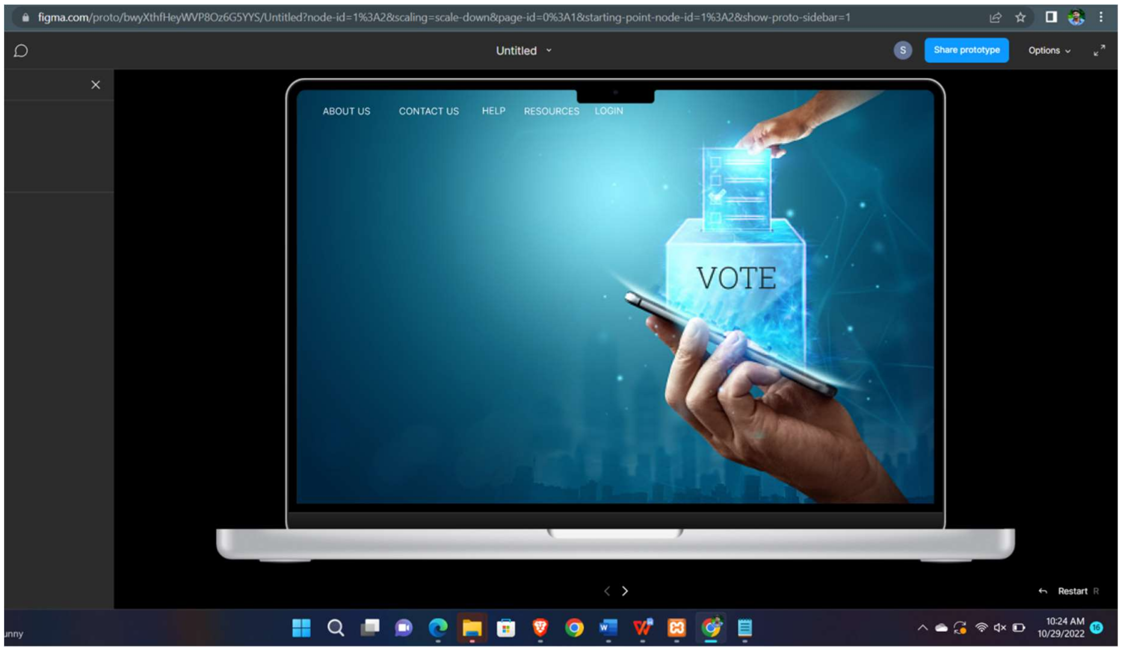
- The object is represented by specifying their name and class.
- It is not mandatory for every class to appear.
- A class may constitute more than one object.
- In the collaboration diagram, firstly, the object is created, and then its class is specified.
- To differentiate one object from another object, it is necessary to name them.

  - Actors: In the collaboration diagram, the actor plays the main role as it invokes the interaction. Each actor has its respective role and name. In this, one actor initiates the use case.
  - Links: The link is an instance of association, which associates the objects and actors. It portrays a relationship between the objects through which the messages are sent. It is represented by a solid line. The link helps an object to connect with or navigate to another object, such that the message flows are attached to links.
  - Messages: It is a communication between objects which carries information and includes a sequence number, so that the activity may take place. It is represented by a labeled arrow, which is placed near a link. The messages are sent from the sender to the receiver, and the direction must be navigable in that particular direction. The receiver must understand the message.

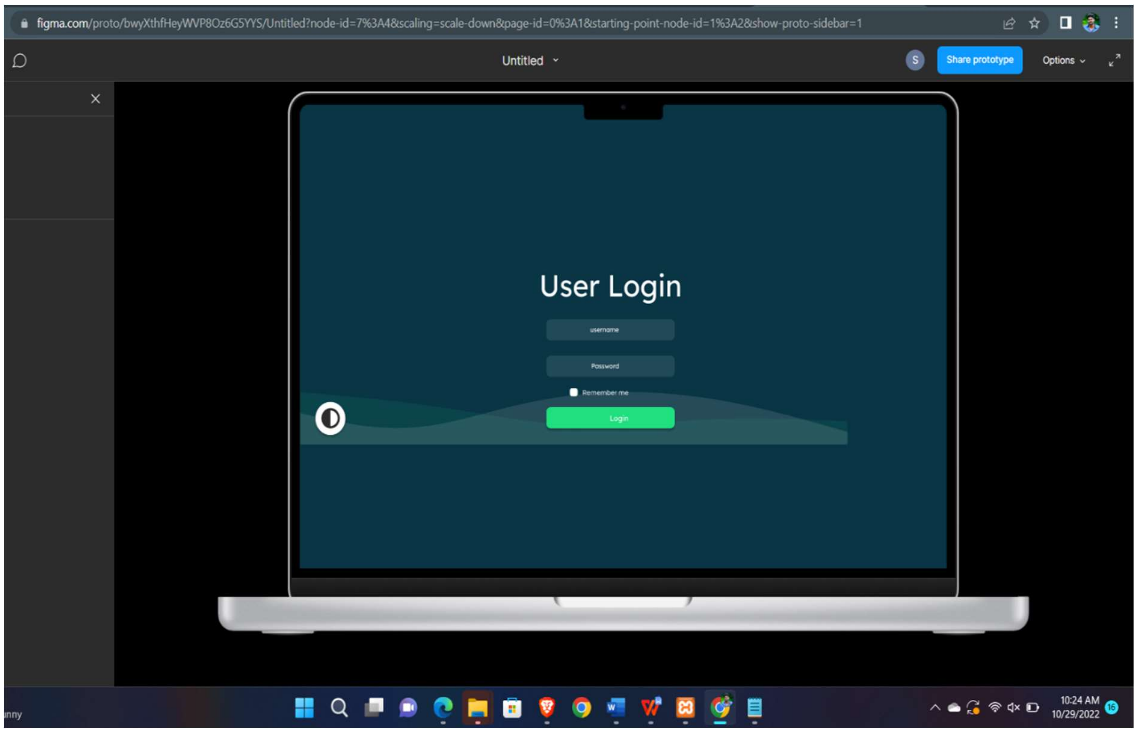

1.login
2.Manage Candidate
3.Manage Subadmin
4.Manage Election
5.Manage Notification
6.Manage Profile
7.Manage History
:Admin
:Server
1.Authentication
2.Candidate details
3.Subadmin details
4.Election
5.Notification
6.Profile
7.History Log
1.Authentication
2.Candidate details
3.Subadmin details
4.Election
5.Notification
6.Profile
7.History Log
1.Aknowledgement
2.Aknowledgement Candidate details
3.Aknowledgement Subadmin details
4.Aknowledgement Election
5.Aknowledgement Notification
6.Aknowledgement Profile
7.Aknowledgement History Log
:Database

4.3 USER INTERFACE DESIGN USING FIGMA

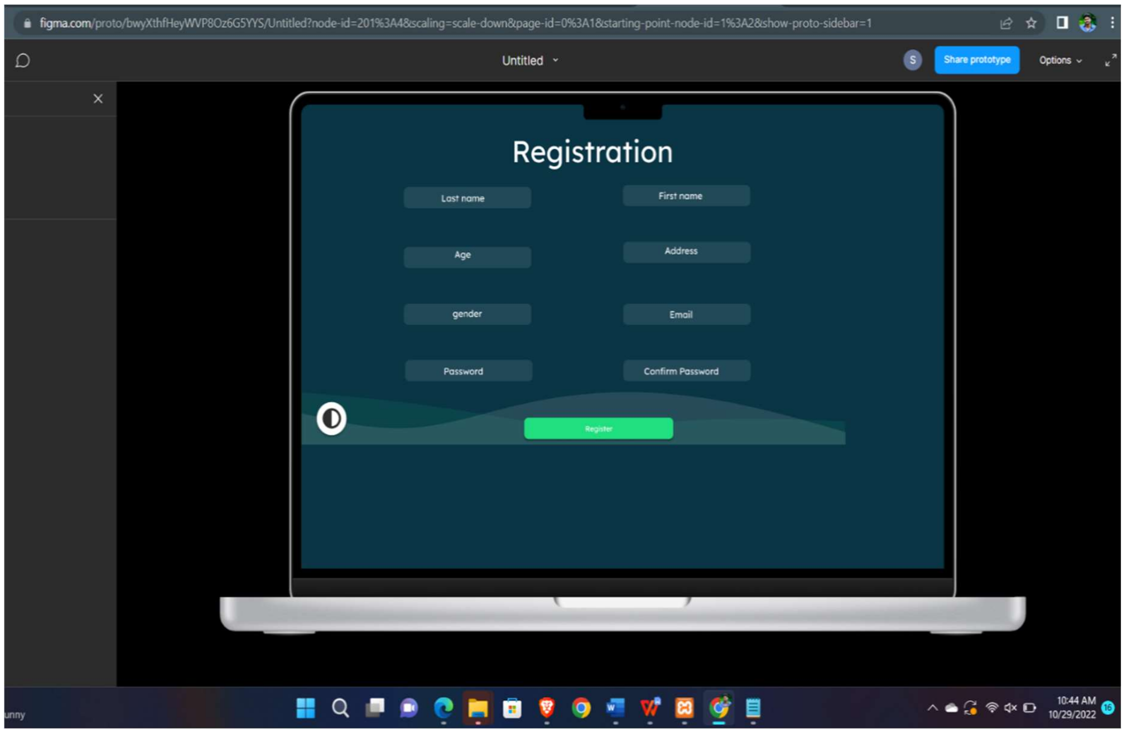
Index page



Login



Registration


figma.com/proto/bwyXthfHeyWVP8Oz6G5YYS/Untitled?node-id=201%3A4&scaling=scale-down&page-id=0%3A1&starting-point-node-id=1%3A2&show-proto-sidebar=1
Untitled
Share prototype
Options
Registration
Last name
First name
Age
Address
gender
Email
Password
Confirm Password
Register
10:44 AM
10/29/2022