**CSE 311 SOFTWARE ENGINEERING AND PROJECT MANGEMENT**

**PROJECT REPORT**

**Election – System**

**Group-16**

**Team Members :**

**Shaik Ali 2022BCS0118 Backend Designer**

**Shoury K 2022BCS0217 Frontend Developer**

**Manoj Kumar 2022BCS0184 Database Architect**

**Yeshwanth I 2022BCD0055 Tester**

**Introduction :**

The **Online Election System** is a web-based application designed to facilitate and streamline the election process by enabling voters to cast their votes online securely and efficiently. The system aims to provide a user-friendly platform for both voters and administrators, ensuring transparency, accuracy, and integrity throughout the entire election process.

**Problem Statement :**

The online election system project addresses the need for a secure, efficient, and accessible digital voting platform to streamline the electoral process. Traditional paper-based voting methods are susceptible to inefficiencies and fraud, whereas an online system can enhance transparency and ease of access for eligible voters. This project aims to develop a robust system with secure user authentication, real-time vote recording, and stringent data integrity measures. Utilizing a PostgreSQL database, it will manage voter, candidate, and election data while implementing audit logs and notifications to ensure accountability and maintain public trust in the digital voting process.

**Objective / Purpose :**

* Protect voter data with encryption and multi-factor authentication to prevent unauthorized access and maintain vote privacy.
* Streamline voter registration and eligibility verification for quick and easy access to the election platform.
* Implement logs and audit trails to track actions, ensuring accountability and transparency without revealing voter identities.

**Feasibility Study :**

* **Technical Feasibility :** Election – system is developed using prominent web technologies like react, postgres and integrates with fastapi.
* **Economic Feasibility :** The economic feasibility of the online election system ensures that the benefits and efficiencies gained from digitizing the election process outweigh the costs of development, deployment, and maintenance.

The system is economically feasible and if it achieves widespread adoption, offers clear benefits.

* **Operational Feasibility :** The operational feasibility of the online election system assesses whether the platform can be smoothly integrated into existing election processes, is user-friendly for both administrators and voters, and reliably supports secure, efficient, and scalable voting operations.
* **Legal Feasibility :** The legal feasibility of the online election system ensures that the platform complies with electoral laws, data protection regulations, and privacy standards, safeguarding voter rights and election integrity while meeting all legal requirements for secure, valid digital voting.

**Functional Requirements :**

These requirements describe the system's behavior and the specific functions it must perform to support voters, administrators, and candidates during the election process.

**User Registration with Organization Email Verification**

*Requirement*: The system must allow voters to register using Google OAuth with their organization-issued email addresses (e.g., @organization.com). Only email domains associated with the organization should be permitted, ensuring that only eligible organization members can register for the election system.

**User Verification through Google OAuth and Organization ID Check**

*Requirement*: The system should verify user eligibility by ensuring they authenticate through Google OAuth with a valid organization email address. If additional verification is required, the system may prompt for organizational ID information (e.g., studentID) to finalize eligibility.

**Candidate Enrollment:**

*Requirement*: The system must allow eligible users to register as candidates in the election by filling out a candidate application form, including details like name, college ID, department, and position they intend to run for.

**Candidate Verification and Approval :**

* *Requirement*: The system should automatically verify candidates' eligibility based on college policies (such as academic standing, age, or department) and may require additional approval by election administrators.

**Election Management :**

* This module enables administrators to set up and manage election cycles, including defining schedules, approving candidates, and managing voter lists. It supports ballot customization, monitors voting progress, and securely calculates and publishes results. These tools ensure a fair and transparent election process, aligning with college guidelines.

**Non – Functional Requirements :**

**Performance Requirements :**

* **Scalability**: Handle high user traffic, especially on election day
* **Availability**: 99.9% uptime during critical periods.
* **Load Handling**: Support a high number of transactions per second.

**Safety and Security Requirements :**

* **Data Encryption**: Encrypt all sensitive data (e.g., votes, personal information) during transmission and storage.
* **Authentication and Authorization**: Implement strong authentication methods (e.g., multi-factor authentication) for users and also ensure proper access controls and permissions to restrict access based on their roles.

**Software Quality Attributes:**

1. **Performance:**

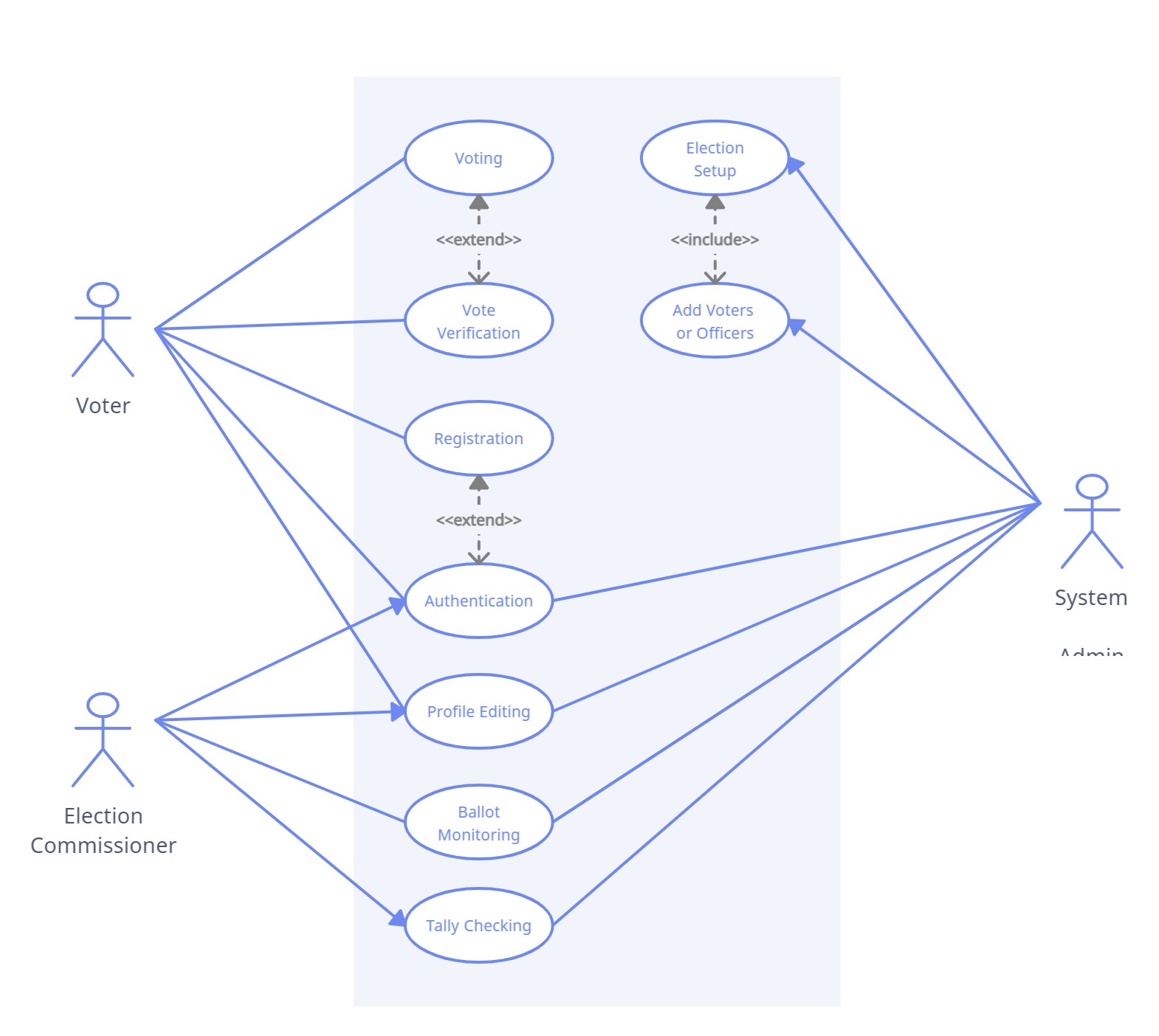
* **Description:** Measures how well the system performs under various conditions, including speed, responsiveness, and resource usage.
* **Examples:** Response time, throughput, and load handling

1. **Reliability:**

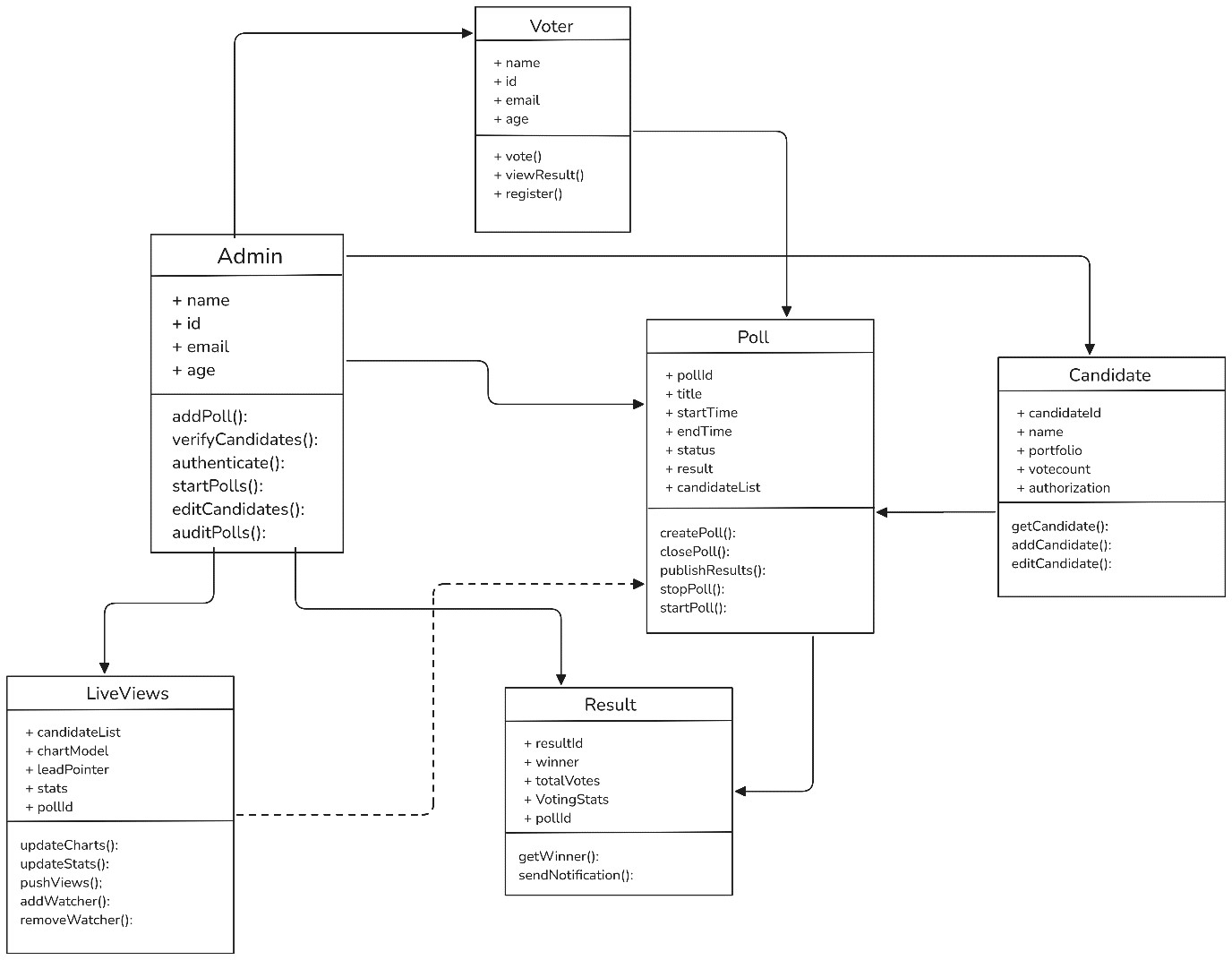
* **Description:** Indicates the system's ability to consistently perform its functions without failure over time.
* **Examples:** System uptime, error rates, and fault tolerance.

**UML Diagrams :**

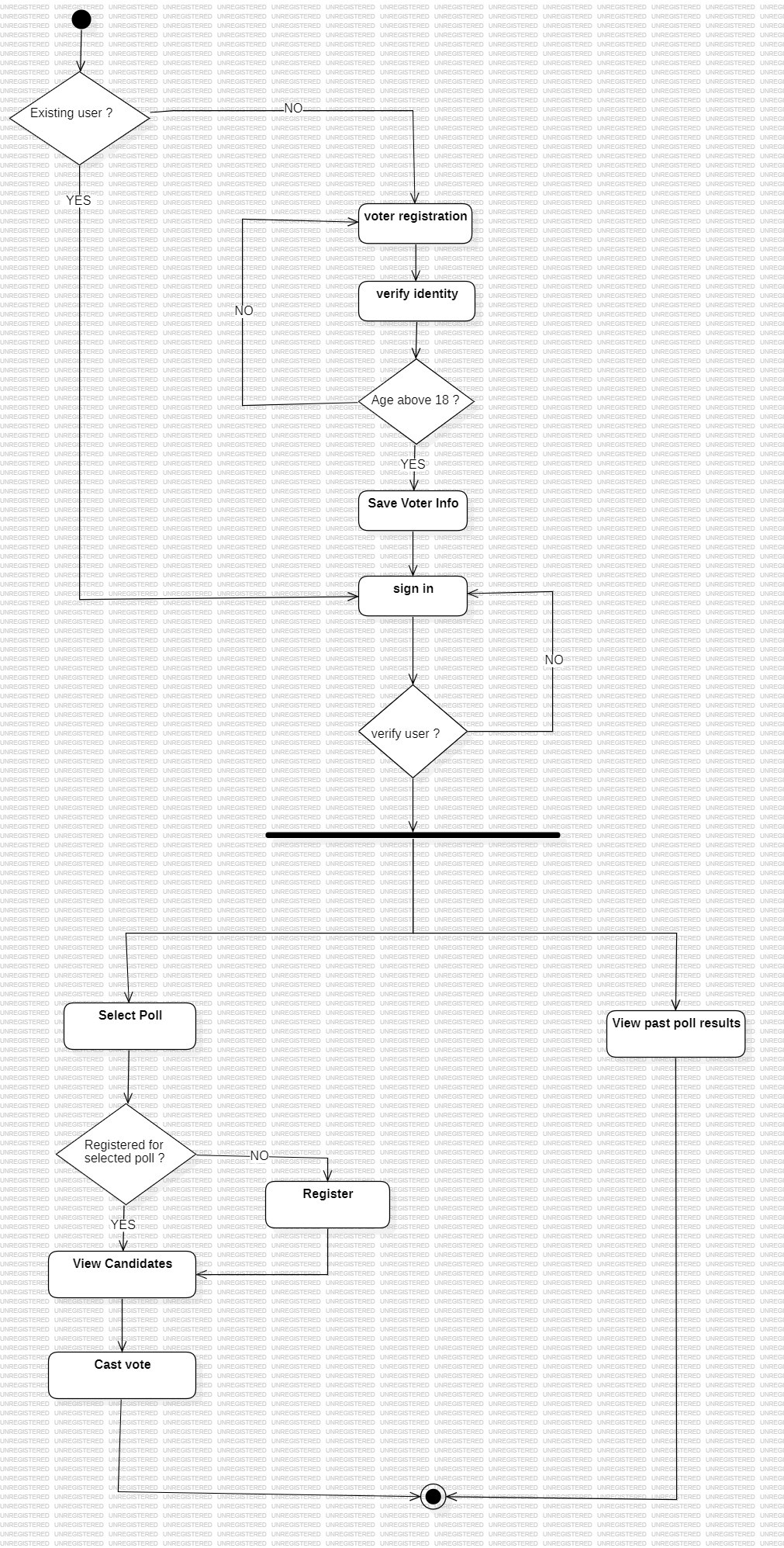
**Use-case Diagram :**

****

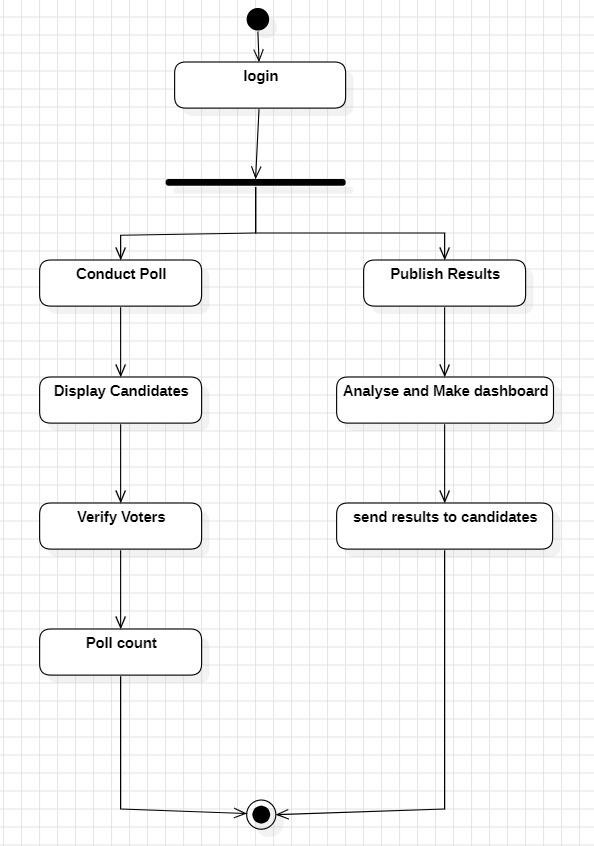
**Class Diagram :**

****

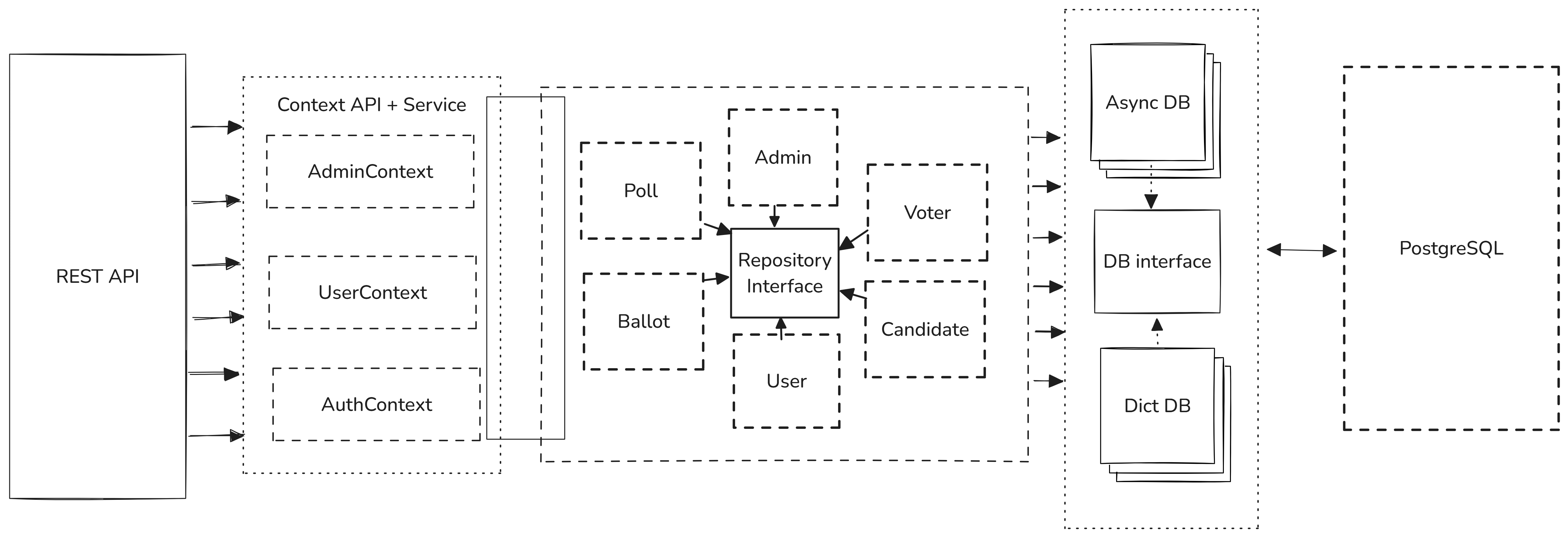
**State chart Diagram :**

****

**Activity Diagram :**

****

**System Architecture :**

****

**Implementation Details :**

* React js : Build Dynamic UI’s faster than ever
* FastApi : Blazing fast API’s, built in flash. FastApi – speed meets simplicity.
* Postgres : Powerful, reliable, open-source. Trust in tour data with PostgreSQL.
* SQLAlchemy : Effortless database interaction,

Python-style.SQLAlchemy – SQL,simplified.

* 12 – Factor App : Code once scale anywhere

**TEST CASE AND REPORTS:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | **Operations** | **Input** | **Expected Output** | **Actual Output** | **Result** |
| **1**  **2**  **3**  **4** | Login | User name and password are taken as input | Navigate to authentication page | Navigates to authentication page | Pass |
| Register | All inputs are given correct where mail id number is primary key | Navigate to login page | Navigates to login page | Pass |
| Join poll | Click | Confirm the user that he/she had successfully joined or not | Confirm the user that he/she had successfully joined or not | Pass |
| Cast vote | Click | Stores the vote and  navigates to thankyou page | Stored the vote and navigates to thankyou page | Pass |
| **4**  **5** | Enter captcha | Captcha | Navigates to voting page | Navigates to voting page | Pass |
| Logout | Click | Logout | Logouts and shows home page | Pass |