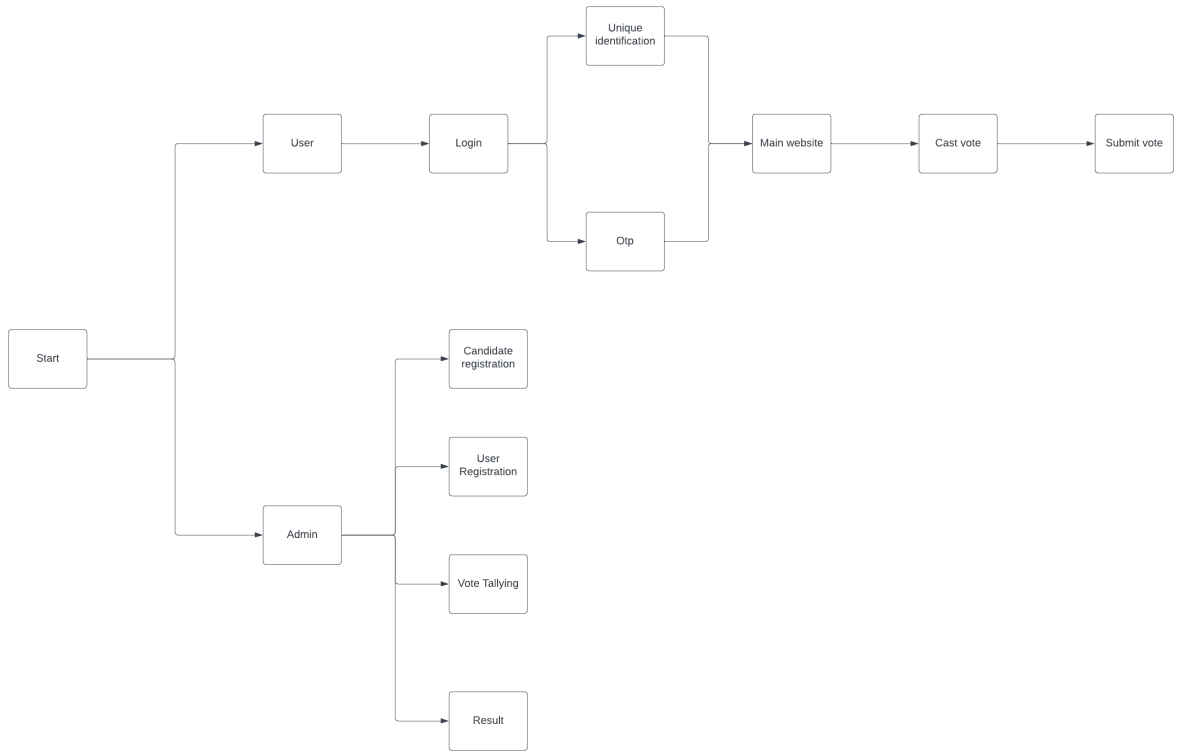
E-Voting Decentralized Application

Software Requirements Specification:



The flow details:

- This project has one profile, and I am considering myself as admin.
- The user will login into the website by using his unique identification number provided by the government of India with the otp for authentication of the user.
- After login, he can cast his vote and to the candidate by user choice anonymously.
- The user can also the live counting of voting.
- The election will complete once every user in the database submitted their votes.
- The user finally sees the winner.
- Here admin manages all the details of candidate information, user information, vote tallying and results.

Project detail steps:

This whole project implementing in AWS cloud, and these are the detail service that will be used for this project.

1. **Amazon EC2**: For deploying the servers that will run the web application and the blockchain nodes.
2. **Amazon DynamoDB**: For storing user registration details and other non-blockchain data.
3. **AWS Lambda**: For running backend services without provisioning servers, such as user registration functions.
4. **AWS Elastic Beanstalk**: For easy deployment and scaling of the web application.
5. **Amazon S3**: For storing and serving static web content.
6. **AWS Amplify**: For building and deploying secure and scalable full-stack applications.
7. **AWS Cognito**: For managing user identities and access, including secure authentication and authorization.
8. **AWS Blockchain Templates**: For launching blockchain networks using popular open-source frameworks.
9. **AWS Key Management Service (KMS)**: For managing cryptographic keys used in the application. (if needed).
10. **AWS CloudWatch**: For monitoring the application's operational health.

Smart Contract Development:

- Deploy your blockchain network using AWS Blockchain Templates with Hyperledger Fabric.
- Write smart contracts for handling voting logic and vote tallying.

Frontend Deployment:

- Use AWS Amplify deploy the frontend user interface (UI) of the application. This frontend can be a web application built using frameworks like Html & CSS.

Voting Process:

- Implement a secure voting API that captures votes and records them on the blockchain.
- Ensure that the API checks for double voting and other fraudulent activities.
- Record each vote as a transaction on the blockchain for immutability and transparency.

Result Tallying:

- Develop smart contract functions that can tally votes securely and transparently.

- Once voting ends, the smart contract should be able to calculate the results and make them available for everyone to verify.

Backend Infrastructure:

- Use EC2 instances or Elastic Beanstalk to host the web application server and blockchain nodes.
- Configure an RDS database to store structured data such as user profiles and DynamoDB for any NoSQL requirements.