

Voter CRM Portal

Capstone Project Presentation

Agenda

- ✓ Overview
- ✓ Features/Functionality
- ✓ Application Flow
- ✓ Tech Stack
- ✓ Frontend
- ✓ Backend
- ✓ Data Science
- ✓ Deployment
- ✓ Testing
- ✓ Demo
- ✓ Team
- ✓ Q&A

Overview

- ✓ The Voter CRM Project arises from the evolving landscape of politics, where traditional methods have given way to data-driven strategies
- ✓ This project draws inspiration from these transformative moments to optimize voter engagement, offering a comprehensive tool designed to empower political actors with data-driven insights and foster lasting connections with the electorate
- ✓ The Voter CRM application is aimed at optimizing voter engagement and campaign management
- ✓ It uses data analytics to empower political campaigns
- ✓ In an era where technology and data play pivotal roles in modern politics, the Voter CRM application stands as a beacon of innovation, offering a user-friendly, secure, and data-driven environment to shape a brighter political landscape

Features/Functionality

- ✓ Admin or Super User login
- ✓ User Registration
- ✓ Subscription Plans
- ✓ Payment Gateway integration
- ✓ User Login (including, Forgot Password or Reset Password)
- ✓ User Groups: MLA Candidate or MP Candidate
 - MLA Candidate user group gets 400 user accounts with default passwords. These 400 users would be the booth coordinators for a given MLA candidate.
 - MP Candidate gets 1000 user accounts
- ✓ Bulk upload of voter data
- ✓ New voter addition to existing database
- ✓ Update voter data
- ✓ Delete voter
- ✓ Dashboard
- ✓ Analytics

Application Flow

- ✓ User Registration -> Choose Subscription Plan (per user group MLA or MP Candidate) -> Payment -> Create User Accounts (400 for MLA Candidate or 1000 for MP Candidate)
- ✓ User Login -> Dashboard -> Analytics
- ✓ User Login -> Voter Maintenance -> View voter data specific to the user group or Bulk upload of voter data or Add/Update/Delete individual voter details or Download voter data
- ✓ Admin or Super User Login -> Access Control for user groups and users

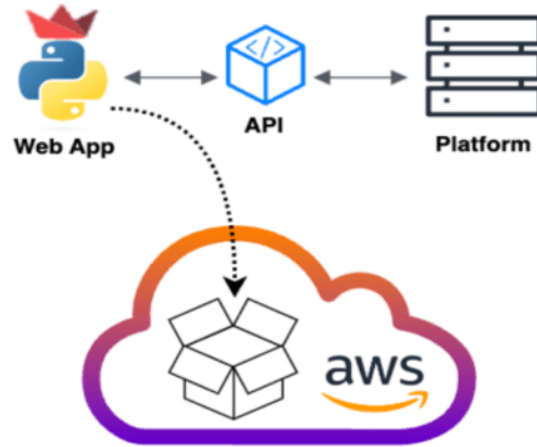
Tech Stack

Technologies/Frameworks/Tools Used



Architecture

Architecture



Frontend

Introduction

- ✓ The frontend is the user-facing interface for managing voter data, opinion polls and campaigns
- ✓ In the realm of voter management and engagement, the frontend of our Voter CRM system plays a pivotal role
- ✓ Our goal is to provide the portal users with a user-friendly yet powerful tool for managing voter relationships, making data-informed decisions, and driving success

Components

- ✓ It's built on Streamlit, a Python library, and employs a modular and component-based approach
- ✓ This architecture ensures real-time interaction and mobile accessibility
- ✓ While Streamlit handles much of the user interface, essential security measures are implemented in the Backend for data protection
- ✓ This robust architecture empowers effective voter management and engagement

Fields

- ✓ The Voter CRM system encompasses a wide range of data fields designed for comprehensive voter information management
- ✓ These fields include unique voter identifiers, personal details, contact information, demographics, preferences, and more
- ✓ Users can efficiently input, manage, and retrieve voter-related data, enabling personalized voter engagement and data-driven decision-making

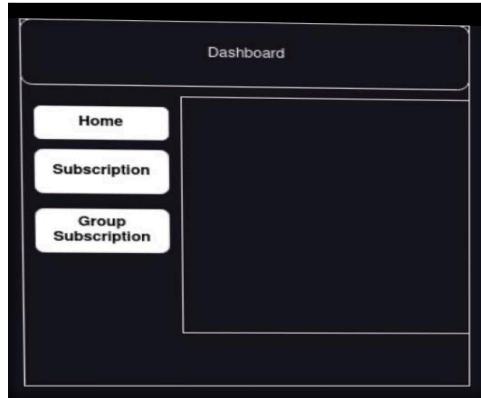
User Registration

- ✓ Users register as "MLA Candidate" or "MP Candidate" with unique usernames
- ✓ They then choose a subscription plan as part of registration and make the payment based on the plan price
- ✓ Once this is completed successfully, an MLA candidate gets 400 user accounts, e.g., mla1 to mla400, with default passwords
- ✓ These user accounts are for the Booth Coordinators who manage voter data, performing tasks like addition, deletion, update, and bulk upload
- ✓ An MP candidate gets 1000 user accounts with default passwords

Subscription Page

- ✓ This section provides users with information about the subscription plans available within Voter CRM portal
 - MLA Monthly
 - MLA Yearly
 - MP Monthly
 - MP Yearly
- ✓ It presents subscription options, pricing, and a step-by-step guide for selecting and upgrading plans
- ✓ Users can choose from various subscription plans, preferred payment methods, and provide necessary billing information for a seamless subscription process

Subscription Page (Contd.)



Voter CRM Group Subscription Page

MLA Monthly	MLA Yearly	MP Monthly	MP Yearly
Rs 15,00,000/Month	Rs 35,00,000/Year	Rs 50,00,000/Month	Rs 75,00,000/Year
Access to 400 members	Access to 400 members	Access to 400 members	Access to 400 members
Subscribe	Subscribe	Subscribe	Subscribe

You have selected Basic Subscription

Please proceed for payment

Credit card

Debit card

Net Banking

UPI

Subscription confirmation

Thank you for your subscription

Transaction ID : Trrdadfkjdf7676234

Plan Name: Basic Subscription

Amount Recieved : Rs 1000

Payment Gateway

- ✓ The Voter CRM portal is integrated with payment gateway - this payment module supports payment methods, billing processes, and the steps users need to follow to complete secure transactions within the portal
- ✓ Users can select their preferred payment method, provide billing information, and receive confirmation upon a successful transaction

Login Page

- ✓ The login page serves as the gateway to the Voter CRM system, ensuring secure access
- ✓ Users are required to enter their unique username and password for authentication
- ✓ This security measure safeguards sensitive voter data and ensures authorized access to the Voter CRM system
- ✓ Users also get an option to reset password
- ✓ Options are provided to the users to recover their password in case they have forgotten their password



The screenshot displays a web interface for a login page. At the top, the title "Login Page" is shown in a large, bold, blue font. Below the title, there are two input fields. The first field is labeled "Username" in a small, blue font and is followed by a light gray rectangular input box. The second field is labeled "Password" in a small, blue font and is followed by a light gray rectangular input box. To the right of the password input box, there is a small blue icon of an eye, which typically represents a toggle for password visibility. Below the password input box, there is a button labeled "Login" in a small, blue font, enclosed in a light gray rectangular box.

Voter Information Form

- ✓ The Voter Information Form enables quick single-entry or Excel upload, validating data, and ensuring swift submission

 **Voter Information Form**

Choose an option:

☒ Input Single Voter Data

☐ Upload Excel File

Voter Number (Integer):

Voter Name (Alphabets and space):

Father/Husband Name (Alphabets and space):

Address (String):

Pin Code (6 digit integer):

Voter Information Form (Contd.)

Voter Information Form

Choose an option:

- ☐ Input Single Voter Data
- ☒ Upload Excel File

Choose an Excel file



Drag and drop file here

Limit 200MB per file • XLSX

Browse files

Process File

Submit

Data validation failed. Please check the following issues: Invalid age value: 16. Age should be greater than 18.

Backend

Backend - Services

✓ Services

- User: User Account Creation, Retrieval, Update, Delete, Validate User
- Subscription: Subscription Plans Creation, Retrieval, Update, Delete
- Audit: Logs the user journey throughout the portal
- Analytics: Runs and retrieves the relevant analytics for the user group of the logged in user

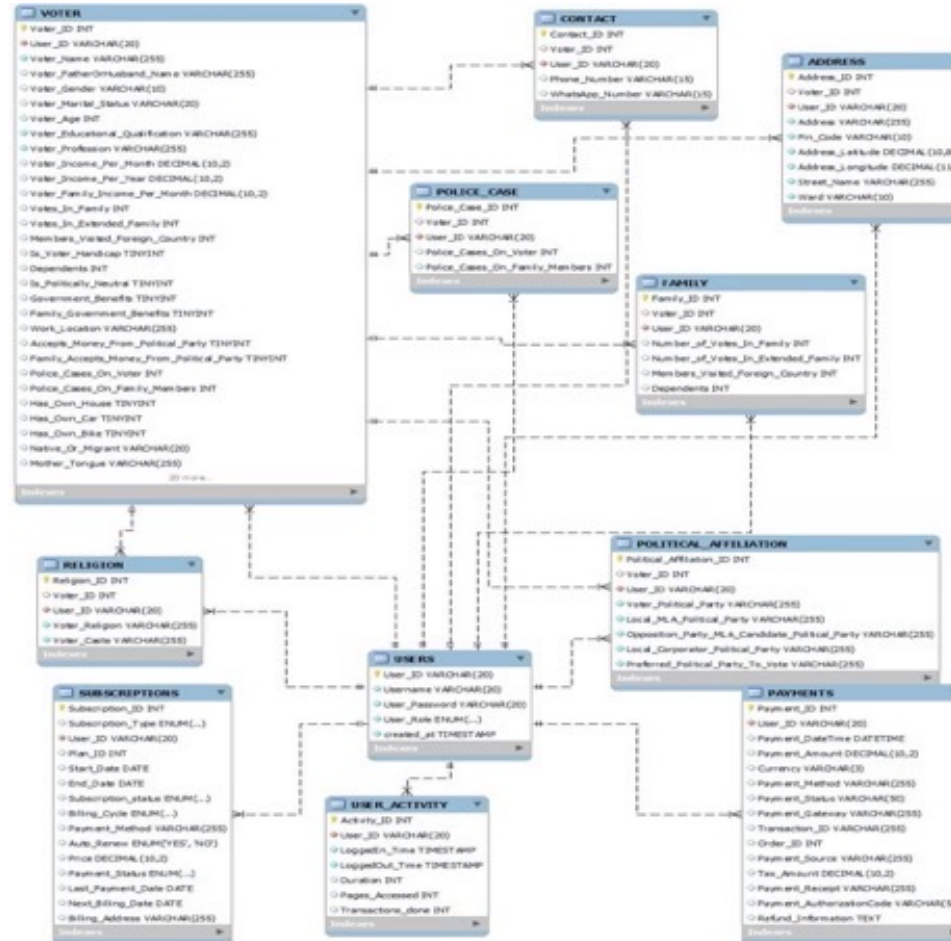
Backend - Services - User Service

- ✓ User Service - validateUser
 - Get the input parameters - username and password from the UI (Frontend)
 - Convert the input to JSON format (jsonify).
 - Redirect it to the URL with the jsonified inputs and call the concerned User Service using requests.post .
`username, password = request.json["Username"], request.json["Password"]`
 - Use MySQL connection to retrieve the data from the USERS table and then do the following steps for validation:
 - Validate the given username and password (input parameters) with the retrieved records to check if there is any match
 - If there is a match found for the given input, this indicates success
 - Return the MySQL query result

Backend - Services - User Service (Contd.)

- Another module will be invoked for the response creation based on the returned jsonified query result from MySQL.
- `response = requests.get(API_URL)` where `API_URL` is `'http://localhost:8501'`.
- Check if the response status is 200 for successful API call.
- If success, will send the Validation success/failure message back to the User Interface (Frontend login page) with `response.txt`.

Backend - Database



Backend - Database (Contd.)

✓ Tables

- User_Group: userGroupId, userGroupName
- Users: User Data - userId, username, password, userGroup, etc.
- Subscriptions: Details of the available subscription plans
 - MLA Monthly plan (Price: Rs. 15,00,000/-)
 - MLA Yearly plan (Price: Rs. 35,00,000/-)
 - MP Monthly plan (Price: Rs. 50,00,000/-)
 - MP Yearly plan (Price: Rs. 75,00,000/-)
- User_Subscription: Stores subscription plans the users have chosen
- User_Activity: User activity from the portal
- Payments: Transaction details of the payments made by the users in the portal

Backend - Database (Contd.)

✓ Tables

- Voter: Data of the voters in the constituency
- Contact: Contact information of the voters
- Address: Address of the voters
- Family: Family specific voter statistics
- Religion: Religion and Case of the voters
- Police_Case: Details of police cases, if any, on the voters or their family members
- Political_Affiliation: Political party affiliation of the voters

Backend - Database (Contd.)

Table 1: Users

- **User_ID**: Unique identifier for users.
- **Username**: User's username (max length: 20).
- **Password**: User's password (max length: 20).
- **Timestamp**: Timestamp of user-related activities.

Table 2: Voter

- **Voter_ID**: Unique identifier for voters (primary key).
- **User_ID**: Reference to the Users table.
- Various voter details, such as name, gender, age, etc.
- **Primary Key**: Voter_ID.
- **Foreign Keys**: Voter_ID references Voter(Voter_ID), User_ID references Users(User_ID).

Table 3: Address

- **Address_ID**: Unique identifier for addresses (primary key, auto-increment).
- **Voter_ID**: Reference to the Voter table.
- **User_ID**: Reference to the Users table.
- Various address details, including latitude and longitude.
- **Primary Key**: Address_ID.
- **Foreign Keys**: Voter_ID references Voter(Voter_ID), User_ID references Users(User_ID).

Backend - Database (Contd.)

Table 4: Contact

- **Contact_ID**: Unique identifier for contacts (primary key, auto-increment).
- **Voter_ID**: Reference to the Voter table.
- **User_ID**: Reference to the Users table.
- Phone numbers and WhatsApp numbers.
- **Primary Key**: Contact_ID.
- **Foreign Keys**: Voter_ID references Voter(Voter_ID), User_ID references Users(User_ID).
- **Constraints**: Unique constraints can be applied to ensure phone numbers or WhatsApp numbers are unique.

Table 5: Family

- **Family_ID**: Unique identifier for families (primary key, auto-increment).
- **Voter_ID**: Reference to the Voter table.
- **User_ID**: Reference to the Users table.
- Family-related details, including votes and dependents.
- **Primary Key**: Family_ID.
- **Foreign Keys**: Voter_ID references Voter(Voter_ID), User_ID references Users(User_ID).

Table 6: Religion

- **Religion_ID**: Unique identifier for religions (primary key, auto-increment).
- **Voter_ID**: Reference to the Voter table.
- **User_ID**: Reference to the Users table.
- Voter's religion and caste details.
- **Primary Key**: Religion_ID.
- **Foreign Keys**: Voter_ID references Voter(Voter_ID), User_ID references Users(User_ID).

Table 7: Political Affiliation

- **Political_Affiliation_ID**: Unique identifier for political affiliations (primary key, auto-increment).
- **Voter_ID**: Reference to the Voter table.
- **User_ID**: Reference to the Users table.
- Various political affiliations, including preferred party to vote.
- **Primary Key**: Political_Affiliation_ID.
- **Foreign Keys**: Voter_ID references Voter(Voter_ID), User_ID references Users(User_ID).

Backend - Database (Contd.)

Table 8: Police Case

- **Police Case ID:** Unique identifier for police cases (primary key, auto-increment).
- **Voter ID:** Reference to the Voter table.
- **User ID:** Reference to the Users table.
- Details about police cases on the voter and their family members.
- **Primary Key:** Police_Case_ID.
- **Foreign Keys:** Voter_ID references Voter(Voter_ID), User_ID references Users(User_ID).

Table 9: Payments

1. **Payment ID:** Unique identifier for payments (primary key, auto-increment).
2. **Voter ID:** Reference to the Voter table.
3. **User ID:** Reference to the Users table.
4. Various payment-related details.
5. **Primary Key:** Payment_ID.
6. **Foreign Keys:** Voter_ID references Voter(Voter_ID), User_ID references Users(User_ID).

Table 10: Subscription

1. **Subscription ID:** Unique identifier for subscriptions (primary key, auto-increment).
2. **User ID:** Reference to the Users table.
3. Details about subscription type, plan, billing, and payment status.
4. **Primary Key:** Subscription_ID.
5. **Foreign Keys:** User_ID references Users(User_ID).

Table 11: User Activity

3. **ID:** Unique identifier for user activities (primary key, auto-increment).
4. **User ID:** Reference to the Users table.
5. Details about user log-in and log-out activities.
6. **Primary Key:** ID.
7. **Foreign Keys:** User_ID references Users(User_ID).

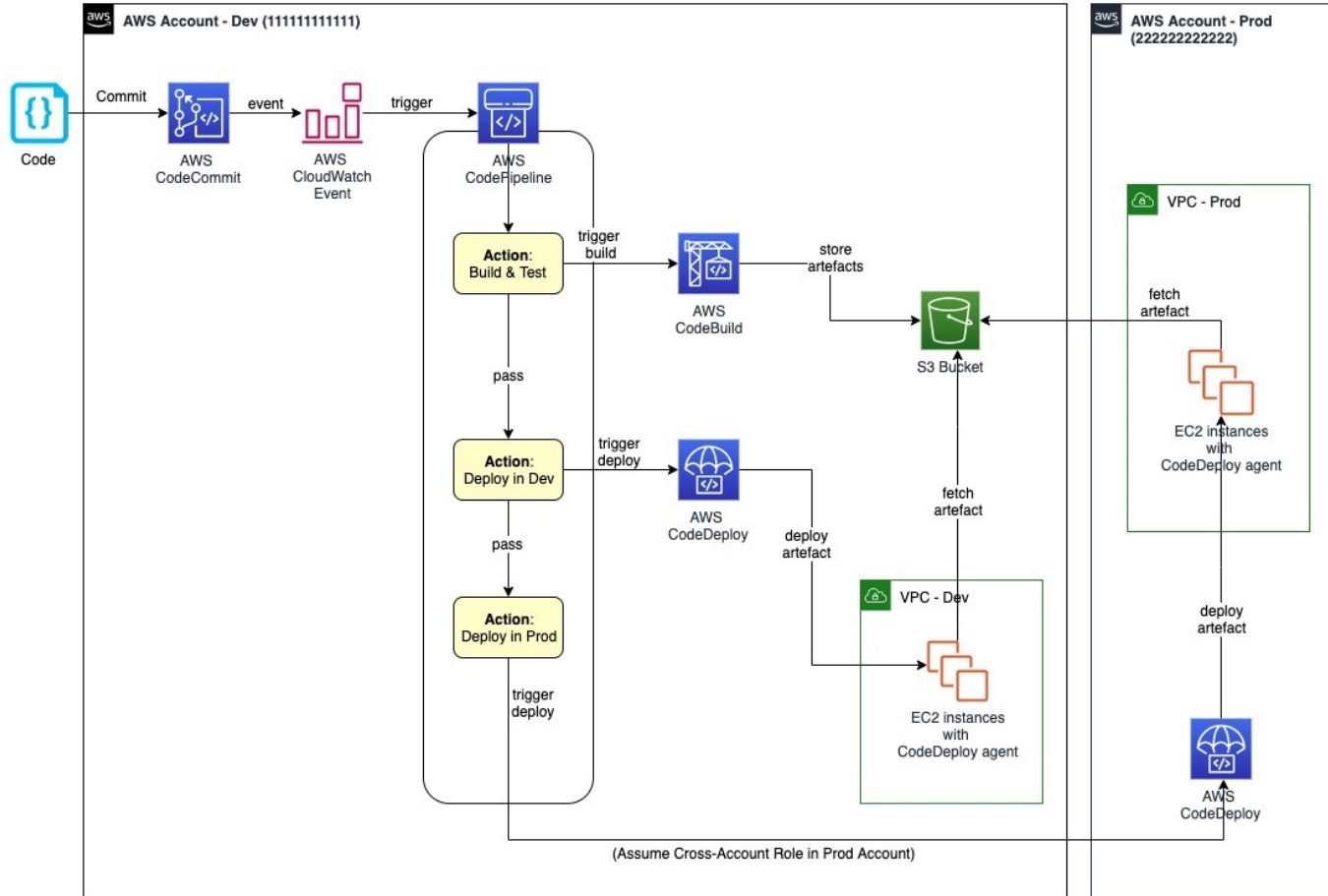
Data Science

Data Analytics/Science

- ✓ Offer insights into the usage patterns of the portal users
- ✓ Analyze user activity and transactions
- ✓ Enable users to gauge the effectiveness of their voter engagement strategies
- ✓ Identify potential red flags, e.g., data leakage by the users to opposition parties
- ✓ Identify patterns in voter data
- ✓ Cluster identification in given voter database
- ✓ Religious groups identification
- ✓ Political party affiliation
- ✓ Sentiment analysis
- ✓ Opinion polls
- ✓ Usage analytics

Deployment

CI/CD Architecture



CI/CD Process

- ✓ **Commit Stage:** The process starts with a code commit to AWS CodeCommit in the Dev account. This triggers an event that is captured by AWS CloudWatch Events.
- ✓ **Build and Test Stage:** AWS CodePipeline is then triggered to orchestrate the build and deployment processes. The first action in the pipeline is to build and test the code using AWS CodeBuild, which upon successful completion, stores the built artifacts in an S3 bucket.
- ✓ **Deployment Stages:**
 - **Dev:** If the build and test stages pass, AWS CodeDeploy automates the deployment of the application to a set of EC2 instances within a Dev Virtual Private Cloud (VPC). The EC2 instances must have the CodeDeploy agent installed.
 - **Prod:** Upon a successful deployment in Dev, the pipeline has an action to deploy in Prod. This action assumes a cross-account role to access the Prod account and triggers AWS CodeDeploy to deploy the artifacts to EC2 instances within the Prod VPC.

Deployment Pipeline - Workflow

✓ **Step 1: Code Commit**

A developer makes changes to the code and commits these changes to a repository in AWS CodeCommit within the Dev AWS Account.

✓ **Step 2: Event Trigger**

The commit triggers an event that AWS CloudWatch Events captures, signaling an update in the source code.

✓ **Step 3: Start Pipeline**

AWS CloudWatch Events notifies AWS CodePipeline, which is configured to listen for such events, and the pipeline's execution starts.

Deployment Pipeline - Workflow (Contd.)

✓ Step 4: Build and Test

- **CodePipeline Action 1:** Initiates AWS CodeBuild, which runs predefined build and test commands (defined in a buildspec file). This environment is scalable and managed, meaning it can adjust to the demands of the build process.
- **Artifact Creation:** Upon a successful build and test process, CodeBuild packages the build output (artifacts) such as compiled code, libraries, and application binaries.
- **Build and Test Stage:** AWS CodePipeline is then triggered to orchestrate the build and deployment

✓ Step 5: Artifact Storage

The artifacts are then stored in a specified Amazon S3 bucket, which serves as a repository for these build artifacts, ready for deployment.

Deployment Pipeline - Workflow (Contd.)

✓ Step 6: Deployment to Development

- **CodePipeline Action 2:** AWS CodeDeploy is triggered, which automates the deployment of the application to a fleet of EC2 instances within the Dev environment's VPC.
- **Deployment Process:** The EC2 instances have the AWS CodeDeploy agent installed, which pulls the deployment artifact from the S3 bucket and deploys it to the instance.

✓ Step 7: Production Deployment Preparations

- **CodePipeline Action 3:** After successful deployment in the Dev environment, a manual or automated approval triggers the deployment process to the Production environment.
- **Assume Cross-Account Role:** Before deployment to Prod, the pipeline assumes a cross-account role, which provides it with the necessary permissions to access resources in the Prod AWS Account.

Deployment Pipeline - Workflow (Contd.)

✓ Step 8: Deployment to Production

- **Prod Deployment:** Using AWS CodeDeploy in the Prod account, the process fetches the artifacts from the S3 bucket and deploys them to EC2 instances in the Production VPC.
- **Finalization:** The EC2 instances in Prod, similar to Dev, have the CodeDeploy agent installed to handle the deployment process.

AWS Services

✓ [AWS CodeCommit](#)

A fully-managed source control service that hosts secure Git-based repositories. CodeCommit makes it easy for teams to collaborate on code in a secure and highly scalable ecosystem. This solution uses CodeCommit to create a repository to store the application and deployment codes.

✓ [AWS CodeBuild](#)

A fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy, on a dynamically created build server. This solution uses CodeBuild to build and test the code, which we deploy later.

✓ [AWS CodeDeploy](#)

A fully managed deployment service that automates software deployments to a variety of compute services such as Amazon EC2, [AWS Fargate](#), [AWS Lambda](#), and your on-premises servers. This solution uses CodeDeploy to deploy the code or application onto a set of EC2 instances running CodeDeploy agents.

AWS Services (Contd.)

✓ [AWS CodePipeline](#)

A fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. This solution uses CodePipeline to create an end-to-end pipeline that fetches the application code from CodeCommit, builds and tests using CodeBuild, and finally deploys using CodeDeploy.

✓ [AWS CloudWatch Events](#)

An AWS CloudWatch Events rule is created to trigger the CodePipeline on a Git commit to the CodeCommit repository.

✓ [Amazon Simple Storage Service \(Amazon S3\)](#)

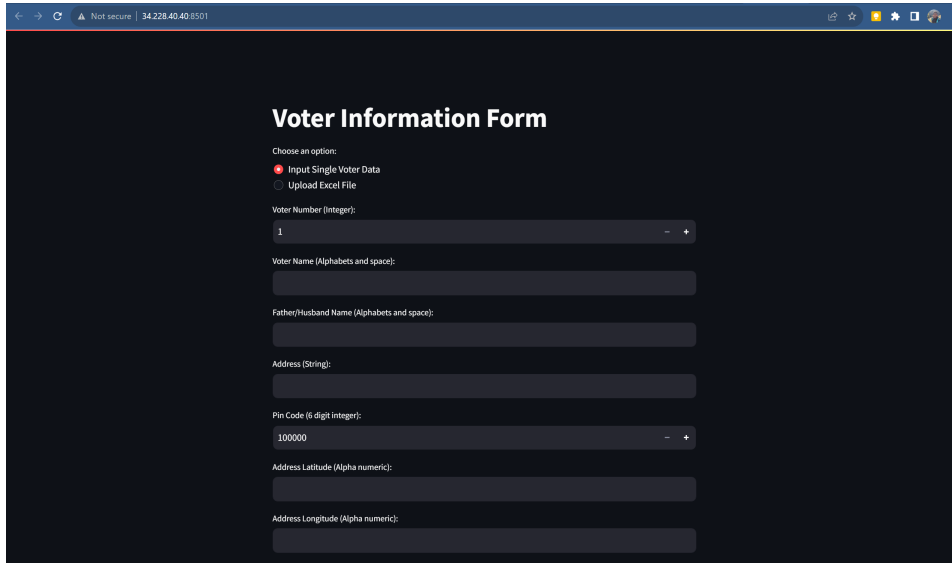
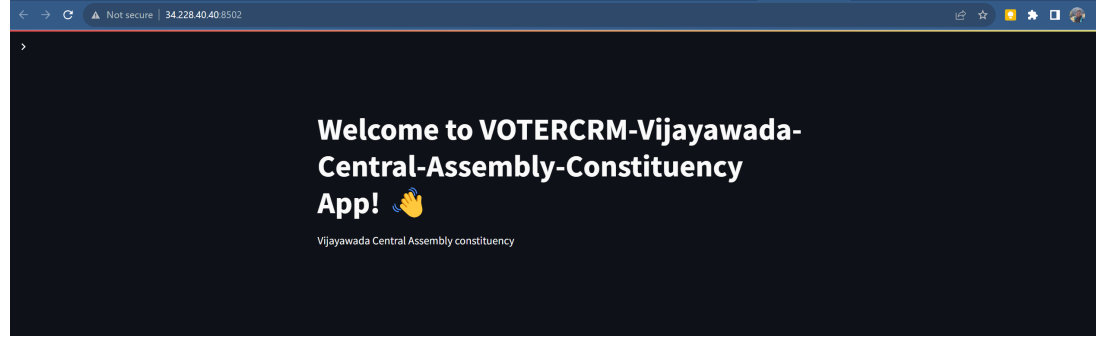
An object storage service that offers industry-leading scalability, data availability, security, and performance. This solution uses an S3 bucket to store the build and deployment artifacts created during the pipeline run.

AWS Services (Contd.)

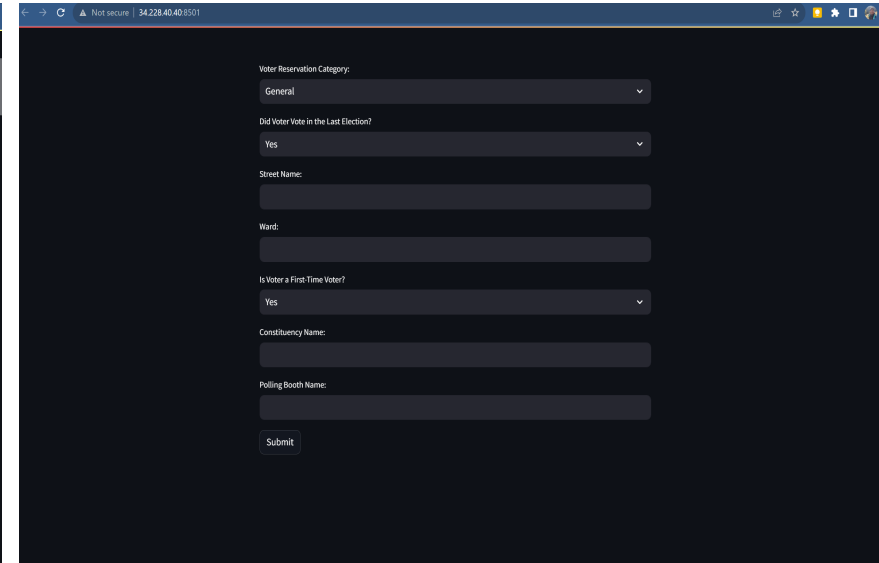
✓ [AWS Key Management Service \(AWS KMS\)](#)

AWS KMS makes it easy for you to create and manage cryptographic keys and control their use across a wide range of AWS services and in your applications. This solution uses AWS KMS to make sure that the build and deployment artifacts stored on the S3 bucket are encrypted at rest.

Application deployed on AWS



A screenshot of the "Voter Information Form" in a web browser. The form is titled "Voter Information Form" and includes a section "Choose an option:" with two radio buttons: "Input Single Voter Data" (selected) and "Upload Excel File". Below this, there are several input fields with labels: "Voter Number (Integer):" with a value of "1", "Voter Name (Alphabets and space):", "Father/Husband Name (Alphabets and space):", "Address (String):", "Pin Code (6 digit integer):" with a value of "100000", "Address Latitude (Alpha numeric):", and "Address Longitude (Alpha numeric):". The browser's address bar shows "Not secure | 34.228.40.40:8501".



A screenshot of the "Voter Reservation Category" form in a web browser. The form includes several dropdown menus and input fields: "Voter Reservation Category:" with a value of "General", "Did Voter Vote in the Last Election?" with a value of "Yes", "Street Name:", "Ward:", "Is Voter a First-Time Voter?" with a value of "Yes", "Constituency Name:", and "Polling booth Name:". A "Submit" button is located at the bottom. The browser's address bar shows "Not secure | 34.228.40.40:8501".

Testing

Testing

- ✓ Software Testing is the process of verifying and evaluating the software or application to ensure that the developed software or product is working as per the requirements

Schema Testing

- ✓ Schema Testing refers to the process of evaluating and validating the structure, format, and organization of a database or data storage system. It involves checking whether the database schema (the blueprint that defines how data is organized) adheres to defined standards, rules, or requirements.

Schema Testing

Test Case 1: Check table presence in a database schema

Steps:

show tables;

Expected Result:

- Table name should be displayed in the list

Actual Result:

✓ As per expectation

Schema Testing

Test Case 2: Check table naming conventions

Steps:

show tables;

Expected Result:

Table name should be a Single Word. Table should not contains spaces.

Actual Result:

✓ As per expectation

Schema Testing

Test Case 2: Check table name conventions

Steps:

- `select count(*) as Number_columns from information_schema.columns where table_name = "address";`

Expected Result:

- This table contain 7 columns

Actual Result:

- As per expectation

Schema Testing

Test Case 3: Check column names in a table

Steps:

- `select column_name from information_schema.columns where table_name = "address"`

Expected Result:

- AddressAddress, IDAddress, LatitudeAddress, LongitudePin, CodeStreet, NameWard

Actual Result:

- As per expectation

Schema Testing

Test Case 4: Check data type of columns in a table

Steps:

- select column_name, data_type from information_schema.columns where table_name = "address"

Expected Result:

Address	varchar
Address_ID	int
Address_Latitude	decimal
Address_Longitude	decimal
Pin_Code	varchar
Street_Name	varchar
Ward	varchar

Actual Result:

- As per expectation

Schema Testing

Test Case 5: Check size of the columns in a table

Steps:

- `select column_name, column_type from information_schema.columns where table_name = "address"`

Expected Result:

Address	varchar(255)
Address_ID	int
Address_Latitude	decimal(10,8)
Address_Longitude	decimal(11,8)
Pin_Code	varchar(10)
Street_Name	varchar(255)
Ward	varchar(10)

Actual Result:

- As per expectation

Schema Testing

Test Case 6: Check null fields in a table

Steps:

- select column_name, is_nullable from information_schema.columns where table_name = "address";

Expected Result:

Address	NO
Address_ID	NO
Address_Latitude	NO
Address_Longitude	NO
Pin_Code	NO
Street_Name	NO
Ward	NO

Actual Result:

- As per expectation

Schema Testing

Test Case 7: Check columns key in a table

Steps:

- `select column_name, column_key from information_schema.columns where table_name = "address";`

Expected Result:

Address	NO
Address_ID	NO
Address_Latitude	NO
Address_Longitude	NO
Pin_Code	NO
Street_Name	NO
Ward	NO

Actual Result:

- As per expectation

Data Mapping Testing

- ✓ Data mapping testing involves validating the accuracy and completeness of data mappings within a system or between systems. It ensures that data moves accurately and appropriately from one source to another, preserving its integrity and meaning.

Data Mapping Testing

Test Case 1: Check new account registration (Create)

Steps:

Admin Validation:

- Open Voter CRM Administration
- Login as administrator
- Go to navigation then click on Voter
- Check the details

Expected Result:

- Voter details should be displayed

Actual Result:

- Voter details are displayed

Data Mapping Testing

Test Case 2: Check details of new account (Retrieve)

Steps:

Admin Validation:

- Open Voter CRM Administration
- Login as administrator
- Go to navigation then click on MP/MLA/User
- Check the user's details

Expected Result:

- Accepted details should be visible

Actual Result:

- As per expectation

Data Integrity Testing

- ✓ Data integrity testing is the process of examining and validating the accuracy, consistency, and reliability of data within a system or a database. Its primary goal is to ensure that data remains intact, consistent, and trustworthy throughout various operations and over time.

Data Integrity Testing

Test Case 1: Check Data Integrity on voter table

Validating: Validate Voter ID

Query: INSERT INTO voter (Voter_ID, Voter_Name, Voter_FatherOrHusband_Name, Voter_Gender, Voter_Marital_Status, Voter_Age, Is_Voter_Handicap, Voter_Educational_Qualification, Voter_Profession, Voter_Income_Per_Month, Voter_Income_Per_Year, Voter_Family_Income_Per_Month, Votes_In_Family, Votes_In_Extended_Family, Members_Visited_Foreign_Country, Dependents, Is_Politically_Neutral, Government_Benefits, Family_Government_Benefits, Work_Location, Accepts_Money_From_Political_Party, Family_Accepts_Money_From_Political_Party, Police_Cases_On_Voter, Police_Cases_On_Family_Members, Has_Own_House, Has_Own_Car, Has_Own_Bike, Native_Or_Migrant, Mother_Tongue, Opinion_On_Present_Government, Opinion_Label_On_Present_Government, Opinion_On_Local_MLA, Opinion_Label_On_Local_MLA, Local_MLA_Political_Party, Opinion_On_Opposition_Party_MLA_Candidate, Opinion_Label_On_Opposition_Party_MLA_Candidate, Opposition_Party_MLA_Candidate_Political_Party, Opinion_On_Local_Corporator_Village_President, Opinion_Label_On_Local_Corporator_Village_President, Local_Corporator_Political_Party, Preferred_Political_Party_To_Vote, BPL, Voter_Monthly_Spending, Voter_Family_Monthly_Spending, Reservation_Category, Voted_In_Last_Election, Voting_First_Time, Constituency_Name, Polling_Booth_Name)
VALUES (1, 'John Doe', 'Robert Doe', 'Male', 'Single', 30, 0, 'Bachelor's Degree', 'Engineer', 5000.00, 60000.00, 15000.00, 4, 6, 2, 2, 1, 1, 0, 'Office', 1, 0, 0, 1, 1, 1, 0, 'Migrant', 'English', 'Satisfied with the government', 'Positive', 'Satisfied with the local MLA', 'Positive', 'Party A', 'Neutral', 'Neutral', 'Party B', 'Satisfied with the local corporator', 'Positive', 'Party C', 'Preferred Party X', 0, 2500.00, 10000.00, 'General', 1, 0, 'Constituency X', 'Polling Booth Y');

Expected Result : Data inserted successfully with valid data

Actual Result : Success

Data Integrity Testing

Test Case 1: Check Data Integrity on voter table

Validating: Validate Voter ID

Query: INSERT INTO voter (Voter_ID, Voter_Name, Voter_FatherOrHusband_Name, Voter_Gender, Voter_Marital_Status, Voter_Age, Is_Voter_Handicap, Voter_Educational_Qualification, Voter_Profession, Voter_Income_Per_Month, Voter_Income_Per_Year, Voter_Family_Income_Per_Month, Votes_In_Family, Votes_In_Extended_Family, Members_Visited_Foreign_Country, Dependents, Is_Politically_Neutral, Government_Benefits, Family_Government_Benefits, Work_Location, Accepts_Money_From_Political_Party, Family_Accepts_Money_From_Political_Party, Police_Cases_On_Voter, Police_Cases_On_Family_Members, Has_Own_House, Has_Own_Car, Has_Own_Bike, Native_Or_Migrant, Mother_Tongue, Opinion_On_Present_Government, Opinion_Label_On_Present_Government, Opinion_On_Local_MLA, Opinion_Label_On_Local_MLA, Local_MLA_Political_Party, Opinion_On_Opposition_Party_MLA_Candidate, Opinion_Label_On_Opposition_Party_MLA_Candidate, Opposition_Party_MLA_Candidate_Political_Party, Opinion_On_Local_Corporator_Village_President, Opinion_Label_On_Local_Corporator_Village_President, Local_Corporator_Political_Party, Preferred_Political_Party_To_Vote, BPL, Voter_Monthly_Spending, Voter_Family_Monthly_Spending, Reservation_Category, Voted_In_Last_Election, Voting_First_Time, Constituency_Name, Polling_Booth_Name)
VALUES (1, 'John Doe', 'Robert Doe', 'Male', 'Single', 30, 0, 'Bachelor's Degree', 'Engineer', 5000.00, 60000.00, 15000.00, 4, 6, 2, 2, 1, 1, 0, 'Office', 1, 0, 0, 1, 1, 1, 0, 'Migrant', 'English', 'Satisfied with the government', 'Positive', 'Satisfied with the local MLA', 'Positive', 'Party A', 'Neutral', 'Neutral', 'Party B', 'Satisfied with the local corporator', 'Positive', 'Party C', 'Preferred Party X', 0, 2500.00, 10000.00, 'General', 1, 0, 'Constituency X', 'Polling Booth Y');

Expected Result : There is a Duplicate Entry error as the Voter_ID is a unique key.

Actual Result : Success

Frontend Testing - Login Page

- Verify Valid Credentials in Login Page:

Login Page

Username

Admin

Password

.....|



Login

Frontend Testing - Login Page

Steps for validating credentials:

- Enter valid username and password.
- User is successfully logged in.

Upon successful login, user is redirected to main page.

Frontend Testing - Login Page

Steps to check for invalid credentials:

- Enter an incorrect username or password or both.
- Appropriate error message is displayed.

Login Page

Username

Admin

Password

.....



Login

Login failed. Please check your credentials.

Frontend Testing - Voter Information Form

Entering Valid data:

Voter Information Form

Choose an option:

- ☒ Input Single Voter Data
☐ Upload Excel File

Voter Number (Integer):

1234567891 - +

Voter Name (Alphabets and space):

kansnkl

Father/Husband Name (Alphabets and space):

klImasd

Address (String):

lkkmsdkmmskld

Pin Code (6 digit integer):

789456 - +

Address Latitude (Alpha numeric):

22.099396

Address Longitude (Alpha numeric):

85.055595

Phone Number (10 digit integer):

1000000000 - +

WhatsApp Number (10 digit integer):

1000000000 - +

Frontend Testing - Voter Information Form

After entering valid data Excel sheet is downloaded.

ADARSH

Ward:

Chroyasi

Is Voter a First-Time Voter?

No

Constituency Name:

surat

Polling Booth Name:

I P SAWANI

Submit

Excel file downloaded successfully!

Frontend Testing - Voter Information Form

Entering invalid data to check if an error is raised or not.

Here entering invalid data in Address latitude and Address longitude

Pin Code (6 digit integer):

789456

- +

Address Latitude (Alpha numeric):

54654655

Address Longitude (Alpha numeric):

4121.21

Frontend Testing - Voter Information Form

Invalid data prevented from frontend and backend also

Is Voter a First-Time Voter?

No

Constituency Name:

surat

Polling Booth Name:

I P SAWANI

Submit

Data validation failed. Please check the following issues: Address Longitude should contain only alphabets and numbers.

Demo

Team

Active Contributors

- ✓ Akhila Mudireddy
- ✓ Amol More
- ✓ Arunabh Pathak
- ✓ Ayush Saxena
- ✓ DSSVK Bharadwaj
- ✓ Harshwardhan Padile
- ✓ Kaushik Pradhan
- ✓ Kavitha Kothuri
- ✓ Khyati Dewan
- ✓ Lakshmi Kondapuram
- ✓ Madhuri Jain
- ✓ Mrudula Jonnavithula
- ✓ Naseemunnisa M
- ✓ Niranjan Nidadavolu
- ✓ Omkar Jawaji
- ✓ Prashanthi Reddy
- ✓ Ram Reddy
- ✓ Ramgopal Nayak
- ✓ Rohit Garg
- ✓ Sagar Patel
- ✓ Sarita Mahanty
- ✓ Shekar Kaki
- ✓ Spoorthy Reddy
- ✓ Sree Lakshmi Kudumula
- ✓ Vamsi Krishna Alla
- ✓ Vivek Dharmagadi

Questions?

Thank you!