**Thakur Degree College of Science and Commerce**

KANDIVALI (EAST)

MUMBAI

**A**

**PROJECT REPORT**

**ON**

**POLL-KARO A` POLLING WEBAPP**

For

THAKUR COLLEGE OF SCIENCE & COMMERCE

Shyamnarayan Thakur Marg, Thakur Village, Kandivali (E), Mumbai - 400 101

Designed and Developed

BY: **Mr. Vivek Vinod Gupta**

Submitted in partial fulfillment of

Bachelors of Science (Computer Science)

**[UNIVERSITY OF MUMBAI]**

**Thakur Degree College of Science and Commerce**

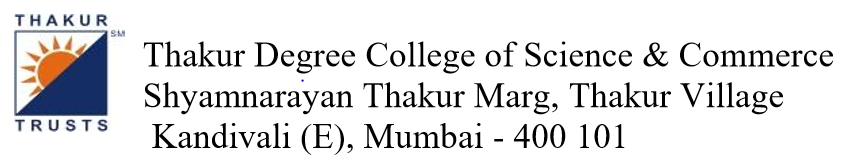
**Kandivali (East), Mumbai.**

**ACADEMIC YEAR 2024-2025**

**CERTIFICATION**

**FROM**

**COLLEGE**



**DATE:**

**COMPUTER SCIENCE DEPARTMENT**

**(2024-2025)**

**Certificate of Approval**

This is to certify that the project work entitled **“Poll-Karo Polling WebApp”**is prepared by **Mr. Vivek Vinod Gupta** a student of **“Third Year Bachelor Of Science (Computer Science)”**course of University of Mumbai, which is conducted by our college.

This is the original study work and important sources used have been duly acknowledged in the report. The report is submitted in partial fulfillment of B.Sc. (Computer Science) course as per rules of University of Mumbai.

**Project Guide Head of Department**

**External Examiner**

**ACKNOWLEDGEMENT**

**ACKNOWLEDGEMENT**

Achievement is finding out what you would be doing rather than what you have to do. It is not until you undertake such a project that you realize how much effort and hard work it really is, what are your capabilities and how well you can present yourself or other things. It tells us how much we rely on the efforts and goodwill of others. It gives me immense pleasure to present this report towards the fulfillment of my project.

It has been rightly said that we are built on the shoulder of others. For everything I have achieved, the credit goes to all those who had helped me to complete this project successfully.

I take this opportunity to express my profound gratitude to management of Thakur Degree College Of Science & Commerce for giving me this opportunity to accomplish this project work.

I am very much thankful to **Mrs.C.T.Chakraborty**- Principal of Thakur College for their kind co-operation in the completion of my project.

A special vote of thanks to my faculty, **Mr. Ashish Trivedi** who is our HOD & also our project guide **Mrs. Sujitha Mohan Mam** for their most sincere, useful and encouraging contribution throughout the project span, without them we couldn’t start and complete the project on time.

Finally, I would like to thank all my friends & entire Computer Science department who directly or indirectly helped me in completion of this project & to my family without whose support, motivation & encouragement this would not have been possible.

**(VIVEK VINOD GUPTA)**

**INDEX**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Index Topic** | **Page** |
|  | **Acknowledgement** | 7 |
| **I.** | **Introduction** |  |
| (i) | **Description of System** | 10 |
| (ii) | **Limitations of present system** | 10 |
| (iii) | **Proposed system and its advantages** | 11 |
| (iv) | **Requirements Specification** | 12 |
| (v) | **Activity Chart** | 14 |
|  |  |  |
| **II** | **System Analysis** | 16 |
| (i) | **Event Table** | 18 |
|  |  |  |
| **III** | **UML Diagram** | 21 |
| (i) | **Use Case Diagram** | 22 |
| (ii) | **Component Diagram** | 23 |
| (iii) | **Deployment Diagram** | 23 |
| (iv) | **State Diagram** | 24 |
| (v) | **Class Diagram** | 25 |
| (vi) | **Collabration Diagram** | 25 |
| (vii) | **Activity Diagram** | 26 |
| (viii) | **Sequence Diagram** | 28 |

|  |  |  |
| --- | --- | --- |
| **IV** | **Data Flow Diagram** |  |
| (i) | **Dataf Flow Diagram(DFD)** | 30 |
|  |  |  |
| **V** | **System Implementation** | 31 |
| (i) | **Code Implementation** | 32 |
|  |  |  |
| **VI** | **Results** | 37 |
| (i) | **Test Cases** | 38 |
| (ii) | **Database Structure** | 41 |
| (iii) | **ScreenShorts** | 46 |
|  |  |  |
| **VII** | **Future Scope** | 51 |
|  |  |  |
| **VIII** | **Conclusion** | 53 |
|  |  |  |
| **IX** | **References and Bibliography** | 55 |

**INTRODUCTION**

**Description of System**

The Polling Web Application is a full-featured online platform designed to create and manage polls efficiently. Built using Django 5.0.7 for the backend and incorporating HTML, CSS, and Bootstrap for the front end, this application allows users to register, create, and participate in polls. Administrators have advanced functionalities to oversee poll activities and user interactions. Users can vote on various polls, and the results are displayed in real-time, enhanced with dynamic data visualization like charts. This system eliminates manual voting processes, ensuring quick access to accurate poll results. User authentication ensures that voting is secure and limited to registered users. Additionally, the application is designed to handle user profiles, providing personalized experiences for each user. The system is scalable, with a responsive and intuitive interface, ensuring usability on both desktop and mobile devices.

**Limitations of Present System**

1. **Manual Record Keeping**: Traditional polling methods often rely on paper-based systems, making it difficult to manage, update, and store data efficiently.
2. **Time-Consuming**: Collecting, counting, and analyzing votes manually takes a significant amount of time, leading to delays in displaying results.
3. **Prone to Errors**: Manual voting processes are susceptible to human errors such as miscounting votes, incorrect entries, or lost records, affecting the accuracy of the results.
4. **Lack of Real-Time Results**: In manual systems, results can only be compiled after the voting process ends, leading to delays, whereas an online system can provide real-time results and analytics.
5. **Limited Accessibility**: Traditional polling systems may require voters to be physically present or use outdated methods, limiting accessibility and convenience for a large audience.

**Proposed System**

The proposed system for the Polling Web Application aims to develop a comprehensive online platform where users can participate in polls seamlessly. This application automates the process of poll creation, voting, and result tracking, providing a user-friendly interface for both regular users and administrators.

Regular users can log in, create polls, vote, and view real-time results, while the admin has the added functionality to manage users, monitor poll activities, and review voting statistics. Built using Django, this system ensures secure authentication, efficient data management, and dynamic chart visualization for poll results.

It also reduces manual efforts and eliminates potential errors associated with traditional voting methods, enhancing the overall efficiency and accuracy of the polling process. The system is scalable and can accommodate future enhancements like advanced analytics and scheduled polls.

**Advantages of Proposed System**

 **Automated Poll Management**: The system automates the entire polling process, reducing the need for manual efforts in creating, managing, and analyzing polls.

 **Real-Time Voting and Results**: Users can vote in real-time, and the results are immediately available for analysis, providing instant feedback and reducing delays in decision-making.

 **Enhanced User Experience**: The platform is user-friendly, allowing users to easily create and participate in polls. The admin interface simplifies poll management and user monitoring.

 **Secure and Efficient Voting System**: The system ensures the security of votes with authentication mechanisms, ensuring only authorized users can vote and preventing vote tampering.

 **Dynamic Analytics Dashboard**: Real-time charts and analytics offer insights into voting patterns, helping users and admins make data-driven decisions.

**REQUIREMENT SPECIFICATION**

**Hardware Requirements**

* + **Processor:** Modern multi-core processor (e.g., Intel Core i5 or equivalent)
  + **RAM:** Minimum of 8 GB
  + **Storage:** SSD with at least 50 GB of free space for development and testing
  + **Network:** Stable internet connection for development, dependency downloads, and version control

**Software Requirements**

* + **Operating System:** Windows, macOS, or Linux
  + **Python Version:** 3.9 or higher
  + **Django Version:** 5.0.7
  + **Database:** PostgreSQL, MySQL, or SQLite (depending on configuration)
  + **Web Browser:** Modern browser for testing (e.g., Google Chrome, Mozilla Firefox)
  + **IDE/Text Editor:** Visual Studio Code, PyCharm, or any preferred code editor

**Front End**

1. **HTML/CSS:** For structuring and styling the web pages
2. **JavaScript:** For interactive elements and dynamic content
3. **Bootstrap:** For responsive design and UI components
4. **Charts Library:** For displaying plots and graphs (e.g., Chart.js )

**Back End**

1. **Django:** Framework for developing the web application
2. **Database:** SQLite
3. **Custom Middleware:** For user and admin access control
4. **Django Apps:** Separate apps for poll creation, user profiles, and analytics

ACTIVITY CHART

THAKUR COLLEGE OF SCIENCE AND COMMERCE

Department of Computer Science

2024-2025

**Student's Name: Gupta Vivek Vinod**

**Project Name: Poll-Karo a Polling WebApp**

**College Name: Thakur College of Science and Commerce.**

|  |  |  |  |
| --- | --- | --- | --- |
| **PHASES** | **EXPECTED DATE OF COMPLETION** | **ACTUALTIMING OF COMPLETION** | **SIGNATURE** |
| Preliminary Investigation |  |  |  |
| System Analysis |  |  |  |
| System Designing |  |  |  |
| System Coding |  |  |  |
| System Implementation |  |  |  |
| Testing Phase |  |  |  |
| Project Submission |  |  |  |

SYSTEM ANALYSIS

EVENT

TABLE

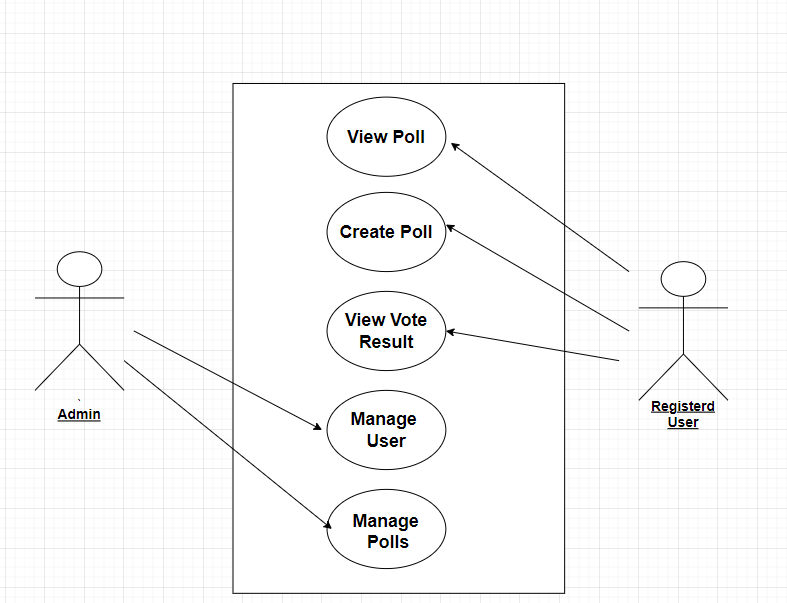
**EVENT TABLE**

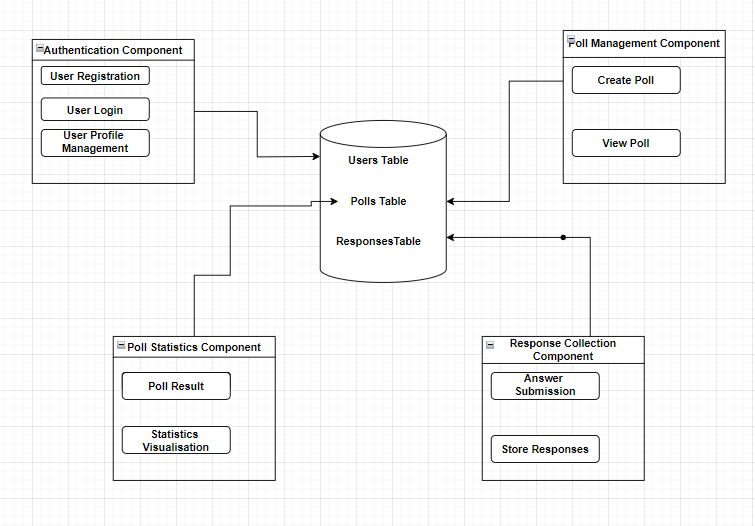
| **Event** | **Trigger** | **Source** | **Response** | **Destination** |
| --- | --- | --- | --- | --- |
| **User Registration** | User submits sign-up form | User | Validate input, create user account | User Profile, Database |
| **User Login** | User submits login form | User | Authenticate user, log in | User Session, Homepage |
| **User Logout** | User clicks logout button | User | End session, log out | Login Page |
| **Create Poll** | User submits poll creation form | User | Validate and store poll data | Database, Poll List |
| **View Polls** | User navigates to homepage or polls page | User | Fetch and display all polls | Homepage |
| **Vote on Poll** | User selects option and submits vote | User | Validate vote, update poll results | Database, Poll Details Page |
| **Comment on Poll** | User submits comment on poll | User | Validate comment, update poll comments | Database, Poll Details Page |
| **View Poll Results** | User clicks on poll result button | User | Fetch poll results | Poll Details Page |
| **Edit Profile** | User submits updated profile information | User | Validate and update profile data | Database, Profile Page |
| **View User Profile** | User navigates to another user's profile page | User | Fetch and display user profile | Profile Page |
| **Analyze Poll Results** | Admin or user views poll results for analysis | User/Admin | Fetch data and display analytics | Analytics Dashboard |

UML

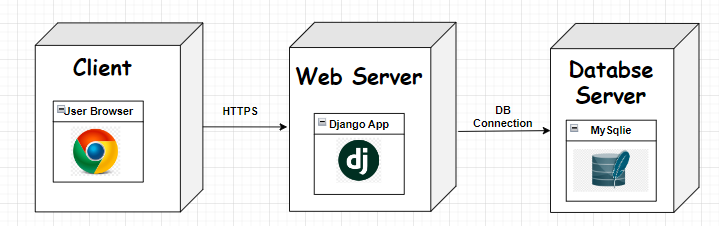
DIAGRAM

**Use Case Diagram**

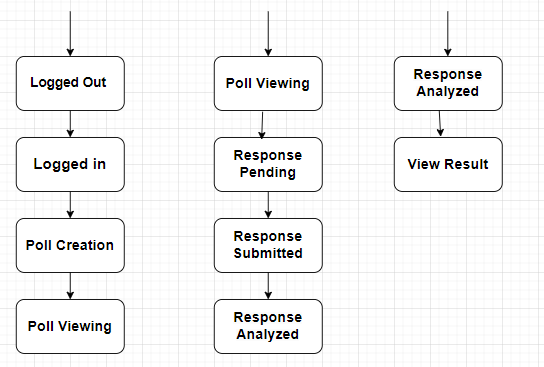


**Component Diagram**

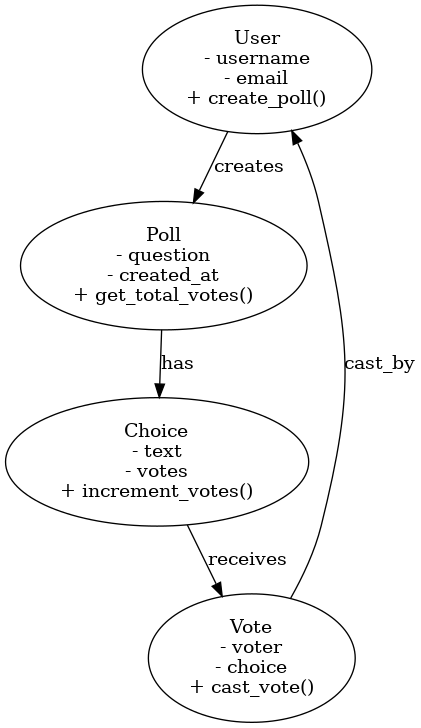
**Deployment Diagram**



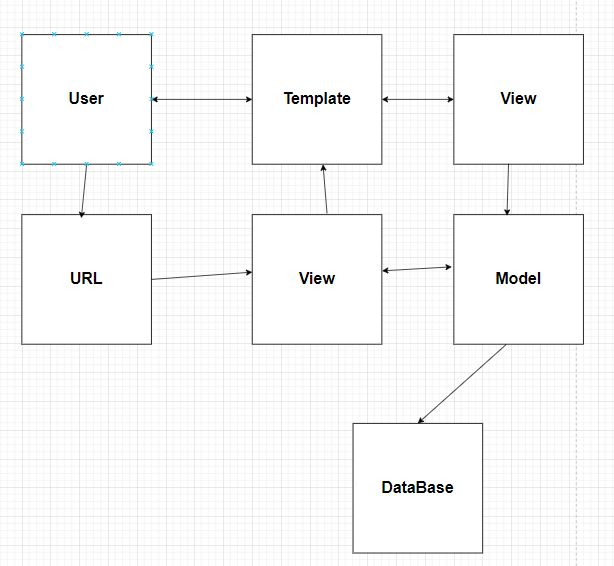
**StateDiagram**



**Class Diagram**

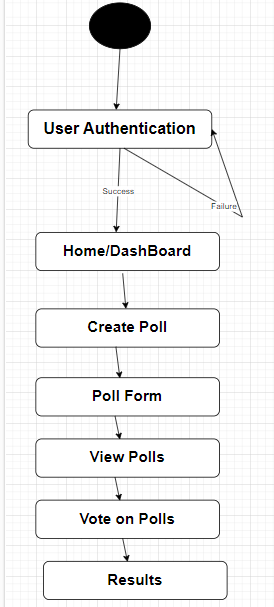


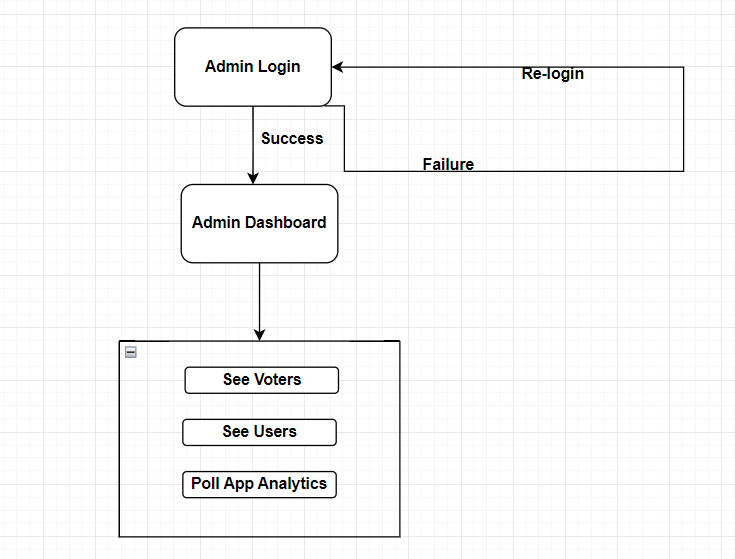
**CollaborationDiagram**



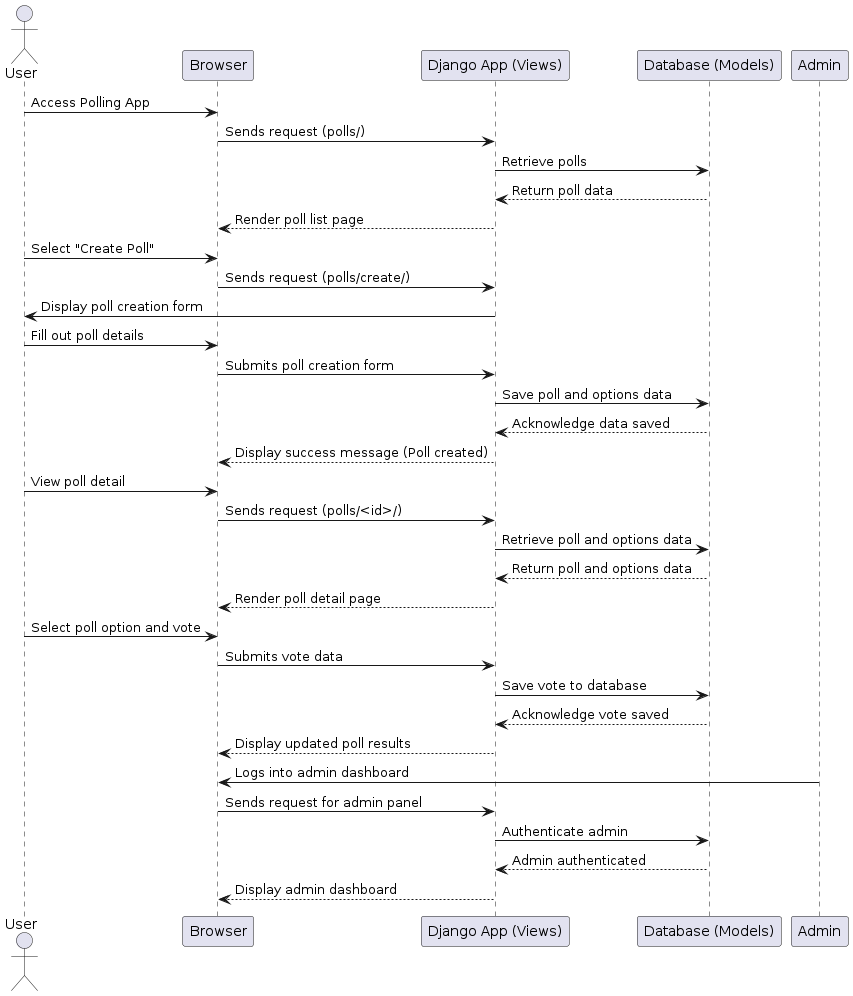
**Activity Diagram**

**REGISTERED USER**



**ADMIN**

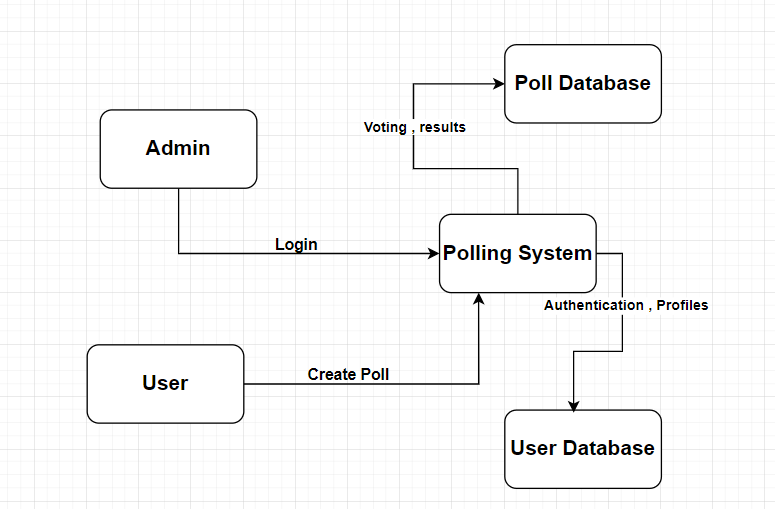
**SequenceDiagram**



Data Flow

Diagram

**Data FlowDiagram(DFD)**



System Implementation

Code Implementation

**HomePage.html**

|  |  |
| --- | --- |
|  | {% extends "base.html" %} |
|  |  |
|  | {% block title %}Welcome to Poll-Karo{% endblock %} |
|  |  |
|  | {% block content %} |
|  | <style> |
|  | .hero-section { |
|  | background: linear-gradient(135deg, #667eea 0%, #764ba2 100%); |
|  | color: white; |
|  | padding: 100px 0; |
|  | margin-bottom: 50px; |
|  | clip-path: polygon(0 0, 100% 0, 100% 85%, 0 100%); |
|  | } |
|  |  |
|  | .hero-title { |
|  | font-size: 3.5rem; |
|  | font-weight: 700; |
|  | margin-bottom: 20px; |
|  | } |
|  |  |
|  | .hero-subtitle { |
|  | font-size: 1.5rem; |
|  | margin-bottom: 30px; |
|  | } |
|  |  |
|  | .recent-polls-heading { |
|  | font-size: 2.5rem; |
|  | font-weight: 600; |
|  | margin-bottom: 30px; |
|  | color: #333; |
|  | text-align: center; |
|  | } |
|  |  |
|  | .poll-card { |
|  | height: 100%; |
|  | transition: all 0.3s ease; |
|  | border: none; |
|  | box-shadow: 0 5px 15px rgba(0,0,0,0.1); |
|  | } |
|  |  |
|  | .poll-card:hover { |
|  | transform: translateY(-10px); |
|  | box-shadow: 0 15px 30px rgba(0,0,0,0.2); |
|  | } |
|  |  |
|  | .card-header { |
|  | background-color: #f8f9fa; |
|  | border-bottom: none; |
|  | } |
|  |  |
|  | .card-title { |
|  | color: #4a4a4a; |
|  | font-weight: 600; |
|  | } |
|  |  |
|  | .card-subtitle { |
|  | font-size: 0.9rem; |
|  | } |
|  |  |
|  | .list-group-item { |
|  | border: none; |
|  | padding: 10px 15px; |
|  | background-color: #f8f9fa; |
|  | margin-bottom: 5px; |
|  | border-radius: 5px; |
|  | transition: background-color 0.3s ease; |
|  | } |
|  |  |
|  | .list-group-item:hover { |
|  | background-color: #e9ecef; |
|  | } |
|  |  |
|  | .btn-floating { |
|  | position: fixed; |
|  | bottom: 30px; |
|  | right: 30px; |
|  | width: 60px; |
|  | height: 60px; |
|  | border-radius: 50%; |
|  | text-align: center; |
|  | font-size: 24px; |
|  | box-shadow: 0 4px 8px rgba(0,0,0,0.1); |
|  | transition: all 0.3s ease; |
|  | z-index: 1000; |
|  | } |
|  |  |
|  | .btn-floating:hover { |
|  | transform: scale(1.1); |
|  | box-shadow: 0 6px 12px rgba(0,0,0,0.2); |
|  | } |
|  |  |
|  | @media (max-width: 768px) { |
|  | .hero-title { |
|  | font-size: 2.5rem; |
|  | } |
|  |  |
|  | .hero-subtitle { |
|  | font-size: 1.2rem; |
|  | } |
|  | } |
|  | </style> |
|  |  |
|  | <div class="hero-section"> |
|  | <div class="container"> |
|  | <h1 class="hero-title">Welcome to Poll-Karo</h1> |
|  | <p class="hero-subtitle">Create, share, and analyze polls with ease</p> |
|  | <a href="[{% url 'create\_poll' %}](file:///C:\Users\User\Desktop\Vivek\Projects\Django\Polling-Webapp\Pole_it\templates\%7b%25%20url%20'create_poll'%20%25%7d)" class="btn btn-light btn-lg">Create Your First Poll</a> |
|  | </div> |
|  | </div> |
|  |  |
|  | <div class="container"> |
|  | <h2 class="recent-polls-heading">Recent Polls</h2> |
|  | <div class="row" id="pollGrid"> |
|  | {% for result in poll\_results %} |
|  | <div class="col-md-4 mb-4"> |
|  | <a href="[{% url 'poll\_results' result.poll.id %}](file:///C:\Users\User\Desktop\Vivek\Projects\Django\Polling-Webapp\Pole_it\templates\%7b%25%20url%20'poll_results'%20result.poll.id%20%25%7d)" class="text-decoration-none"> |
|  | <div class="card poll-card"> |
|  | <div class="card-header"> |
|  | <h5 class="card-title">{{ result.poll.title }}</h5> |
|  | </div> |
|  | <div class="card-body"> |
|  | <h6 class="card-subtitle mb-2 text-muted">{{ result.poll.question }}</h6> |
|  | <ul class="list-group mt-3"> |
|  | {% for option in result.poll.options.all %} |
|  | <li class="list-group-item d-flex justify-content-between align-items-center"> |
|  | {{ option.option\_text }} |
|  | <span class="badge bg-primary rounded-pill">{{ option.votes.count }}</span> |
|  | </li> |
|  | {% endfor %} |
|  | </ul> |
|  | <p class="mb-1 mt-2"><strong><i class="bi bi-person"></i></strong> {{ result.poll.creator.username }}</p> |
|  | <p class="mb-3 mt-2"><strong><i class="bi bi-bar-chart"></i></strong> {{ result.total\_votes }}</p> |
|  | </div> |
|  | </div> |
|  | </a> |
|  | </div> |
|  | {% empty %} |
|  | <div class="col-12 text-center"> |
|  | <p class="lead">No polls available. Be the first to create one!</p> |
|  | <a href="[{% url 'create\_poll' %}](file:///C:\Users\User\Desktop\Vivek\Projects\Django\Polling-Webapp\Pole_it\templates\%7b%25%20url%20'create_poll'%20%25%7d)" class="btn btn-primary btn-lg mt-3">Create a Poll</a> |
|  | </div> |
|  | {% endfor %} |
|  | </div> |
|  | </div> |
|  |  |
|  | <a href="[{% url 'create\_poll' %}](file:///C:\Users\User\Desktop\Vivek\Projects\Django\Polling-Webapp\Pole_it\templates\%7b%25%20url%20'create_poll'%20%25%7d)" class="btn btn-primary btn-floating" id="createPollBtn"> |
|  | <i class="fas fa-plus"></i> |
|  | </a> |
|  |  |
|  | <script src="<https://unpkg.com/masonry-layout@4/dist/masonry.pkgd.min.js>"></script> |
|  | <script> |
|  | document.addEventListener('DOMContentLoaded', function() { |
|  | var elem = document.querySelector('#pollGrid'); |
|  | var msnry = new Masonry(elem, { |
|  | itemSelector: '.col-md-4', |
|  | percentPosition: true |
|  | }); |
|  |  |
|  | var createPollBtn = document.getElementById('createPollBtn'); |
|  | createPollBtn.addEventListener('click', function(e) { |
|  | e.preventDefault(); |
|  | this.classList.add('animate\_\_animated', 'animate\_\_rubberBand'); |
|  | setTimeout(() => { |
|  | window.location.href = this.getAttribute('href'); |
|  | }, 500); |
|  | }); |
|  |  |
|  | var pollCards = document.querySelectorAll('.poll-card'); |
|  | pollCards.forEach(card => { |
|  | card.addEventListener('mouseenter', function() { |
|  | this.style.transform = 'translateY(-10px)'; |
|  | }); |
|  | card.addEventListener('mouseleave', function() { |
|  | this.style.transform = 'translateY(0)'; |
|  | }); |
|  | }); |
|  | }); |
|  | </script> |
|  | {% endblock %} |
|  |  |

**Views.py**

# Authentication/views.py

from django.shortcuts import render, redirect

from django.contrib.auth import authenticate, login

from django.contrib import messages

from django.contrib.auth.models import User

from django.contrib.auth.decorators import login\_required

from Poll.models import Poll

from .models import UserProfile

from .forms import CustomLoginForm

from .forms import UserProfileForm

from .models import UserProfile

def landing\_page(request):

return render(request, 'auth/landingPage.html')

@login\_required

def home\_view(request):

user = request.user

polls = Poll.objects.all()

poll\_results = []

for poll in polls:

options = poll.options.all()

labels = [option.option\_text for option in options]

data = [option.votes.count() for option in options]

total\_votes = sum(data) # Calculate total votes for the poll

poll\_results.append({

'poll': poll,

'labels': labels,

'data': data,

'total\_votes': total\_votes,

})

context = {

'user': user,

'role': 'Superuser' if user.is\_superuser else 'Regular User',

'poll\_results': poll\_results

}

return render(request, 'HomePage.html', context)

def login\_view(request):

if request.method == "POST":

form = CustomLoginForm(request, data=request.POST)

if form.is\_valid():

username = form.cleaned\_data.get('username')

password = form.cleaned\_data.get('password')

is\_admin = form.cleaned\_data.get('is\_admin')

user = authenticate(request, username=username, password=password)

if user is not None and user.is\_active:

# Automatically create UserProfile if it doesn't exist

if not UserProfile.objects.filter(user=user).exists():

UserProfile.objects.create(user=user)

# Admin-specific login

if is\_admin and user.is\_superuser:

login(request, user)

return redirect('analytics\_dashboard') # Redirect to admin dashboard

elif not is\_admin:

login(request, user)

return redirect('home') # Redirect to the home page or user dashboard

else:

messages.error(request, "Username or Password is invalid!") # Handle invalid form submission

else:

form = CustomLoginForm()

return render(request, 'auth/login.html', {'form': form})

def signup\_view(request):

if request.method == "POST":

username = request.POST['username']

email = request.POST['email']

password = request.POST['password']

password\_confirm = request.POST['password\_confirm']

if User.objects.filter(username=username).exists():

messages.error(request, "Username already exists.")

return render(request, 'auth/signup.html')

if password == password\_confirm:

user = User.objects.create\_user(username=username, email=email, password=password)

UserProfile.objects.create(user=user) # Create UserProfile instance during signup

login(request, user)

return redirect('login')

else:

messages.error(request, "Passwords do not match.")

return render(request, 'auth/signup.html')

return render(request, 'auth/signup.html')

@login\_required

def view\_profile(request):

""" View profile details """

user\_profile = UserProfile.objects.get(user=request.user)

context = {'user\_profile': user\_profile}

return render(request, 'auth/view\_profile.html', context)

@login\_required

def edit\_profile(request):

""" Update profile details """

user\_profile = UserProfile.objects.get(user=request.user)

if request.method == 'POST':

form = UserProfileForm(request.POST, request.FILES, instance=user\_profile)

if form.is\_valid():

form.save()

return redirect('view\_profile')

else:

form = UserProfileForm(instance=user\_profile)

context = {'form': form}

return render(request, 'auth/edit\_profile.html', context)

**Create\_poll.html**

|  |  |
| --- | --- |
|  | {% extends "base.html" %} |
|  |  |
|  | {% block title %}Create Poll - Poll-Karo{% endblock %} |
|  |  |
|  | {% block content %} |
|  | <div class="container mt-5"> |
|  | <div class="row justify-content-center"> |
|  | <div class="col-lg-8 col-md-10"> |
|  | <div id="form-card" class="card shadow-lg border-0 animate-card"> |
|  | <div class="card-header bg-gradient text-center py-4 rounded-top"> |
|  | <h2 class="mb-0">Create a New Poll</h2> |
|  | </div> |
|  | <div class="card-body p-5"> |
|  | <form method="post" action="{% url 'create\_poll' %}" enctype="multipart/form-data" id="create-poll-form"> |
|  | {% csrf\_token %} |
|  | <div class="mb-4"> |
|  | <label for="poll-title" class="form-label">Poll Title</label> |
|  | <input type="text" class="form-control form-control-lg" id="poll-title" name="title" placeholder="Enter poll title" required> |
|  | </div> |
|  | <div class="mb-4"> |
|  | <label for="poll-question" class="form-label">Poll Question</label> |
|  | <textarea class="form-control" id="poll-question" name="question" rows="4" placeholder="Enter the poll question" required></textarea> |
|  | </div> |
|  | <div id="options-container" class="mb-4"> |
|  | <label class="form-label">Poll Options</label> |
|  | <div class="option-item mb-2"> |
|  | <div class="input-group"> |
|  | <span class="input-group-text"><i class="bi bi-1-circle"></i></span> |
|  | <input type="text" class="form-control" name="options" placeholder="Option 1" required> |
|  | <button type="button" class="btn btn-outline-danger remove-option"><i class="bi bi-trash"></i></button> |
|  | </div> |
|  | </div> |
|  | <div class="option-item mb-2"> |
|  | <div class="input-group"> |
|  | <span class="input-group-text"><i class="bi bi-2-circle"></i></span> |
|  | <input type="text" class="form-control" name="options" placeholder="Option 2" required> |
|  | <button type="button" class="btn btn-outline-danger remove-option"><i class="bi bi-trash"></i></button> |
|  | </div> |
|  | </div> |
|  | </div> |
|  | <div class="d-flex justify-content-between mb-4"> |
|  | <button type="button" id="add-option" class="btn btn-outline-primary"> |
|  | <i class="bi bi-plus-lg"></i> Add Option |
|  | </button> |
|  | </div> |
|  | <button type="submit" class="btn btn-primary btn-lg w-100 animate-submit">Create Poll</button> |
|  | </form> |
|  | </div> |
|  | </div> |
|  | </div> |
|  | </div> |
|  | </div> |
|  | </div> |
|  | </div> |
|  |  |
|  | <link href="<https://cdn.jsdelivr.net/npm/bootstrap-icons/font/bootstrap-icons.css>" rel="stylesheet"> |
|  |  |
|  | <style> |
|  | :root { |
|  | --primary-color: #4a90e2; |
|  | --secondary-color: #f39c12; |
|  | --success-color: #2ecc71; |
|  | --danger-color: #e74c3c; |
|  | --light-color: #ecf0f1; |
|  | --dark-color: #34495e; |
|  | } |
|  |  |
|  | body { |
|  | background-color: var(--light-color); |
|  | color: var(--dark-color); |
|  | } |
|  |  |
|  | .animate-card { |
|  | transition: transform 0.4s ease, box-shadow 0.4s ease; |
|  | background-color: #ffffff; |
|  | border-radius: 15px; |
|  | } |
|  |  |
|  | .animate-card:hover { |
|  | transform: translateY(-5px); |
|  | box-shadow: 0 15px 30px rgba(0, 0, 0, 0.1); |
|  | } |
|  |  |
|  | .card-header { |
|  | background: linear-gradient(135deg, var(--primary-color), var(--secondary-color)); |
|  | border-bottom: none; |
|  | color: blue; |
|  | } |
|  |  |
|  | .animate-submit { |
|  | transition: all 0.3s ease; |
|  | background-color: var(--primary-color); |
|  | border-color: var(--primary-color); |
|  | } |
|  |  |
|  | .animate-submit:hover { |
|  | background-color: var(--secondary-color); |
|  | border-color: var(--secondary-color); |
|  | transform: translateY(-2px); |
|  | box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1); |
|  | } |
|  |  |
|  | .form-control { |
|  | transition: box-shadow 0.3s ease, border-color 0.3s ease; |
|  | border-radius: 8px; |
|  | } |
|  |  |
|  | .form-control:focus { |
|  | box-shadow: 0 0 0 0.2rem rgba(74, 144, 226, 0.25); |
|  | border-color: var(--primary-color); |
|  | } |
|  |  |
|  | .option-item .input-group-text { |
|  | background-color: var(--primary-color); |
|  | color: white; |
|  | border: none; |
|  | } |
|  |  |
|  | .btn-outline-danger { |
|  | color: var(--danger-color); |
|  | border-color: var(--danger-color); |
|  | } |
|  |  |
|  | .btn-outline-danger:hover { |
|  | background-color: var(--danger-color); |
|  | color: white; |
|  | } |
|  |  |
|  | .btn-outline-primary { |
|  | color: var(--primary-color); |
|  | border-color: var(--primary-color); |
|  | } |
|  |  |
|  | .btn-outline-primary:hover { |
|  | background-color: var(--primary-color); |
|  | color: white; |
|  | } |
|  |  |
|  | .animate-photo-card { |
|  | transition: all 0.4s ease; |
|  | } |
|  |  |
|  | .photo-card-visible { |
|  | opacity: 1; |
|  | transform: translateY(0); |
|  | } |
|  |  |
|  | .photo-card-hidden { |
|  | opacity: 0; |
|  | transform: translateY(-15px); |
|  | } |
|  |  |
|  | #uploaded-photo { |
|  | max-height: 300px; |
|  | object-fit: cover; |
|  | border-radius: 10px; |
|  | box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1); |
|  | } |
|  |  |
|  | @media (max-width: 992px) { |
|  | #form-card, #photo-preview { |
|  | width: 100%; |
|  | } |
|  | } |
|  | </style> |
|  |  |
|  | {% endblock %} |
|  |  |

**Views.py**

from django.shortcuts import render, get\_object\_or\_404, redirect

from .models import Poll, PollOption, Vote

from .forms import PollForm, VoteForm

from django.contrib.auth.decorators import login\_required, user\_passes\_test

from django.db.models import Count

from Authentication.models import User

from django.db.models.functions import TruncDate

def poll\_visualizations\_view(request):

polls = Poll.objects.all()

poll\_data = []

for poll in polls:

options = poll.options.all()

labels = [option.option\_text for option in options]

data = [option.votes.count() for option in options]

poll\_data.append({

'poll': poll,

'labels': labels,

'data': data,

})

return render(request, 'polls/poll\_visualizations.html', {

'poll\_data': poll\_data,

})

@login\_required

def create\_poll\_view(request):

if request.method == "POST":

poll\_form = PollForm(request.POST)

if poll\_form.is\_valid():

poll = poll\_form.save(commit=False)

poll.creator = request.user # Associate poll with the user

poll.save()

options = request.POST.getlist('options') # Get options from the form

for option\_text in options:

PollOption.objects.create(poll=poll, option\_text=option\_text)

return redirect('poll\_list') # Redirect to the poll list after saving

else:

poll\_form = PollForm()

return render(request, 'polls/create\_poll.html', {'poll\_form': poll\_form})

def vote\_view(request, poll\_id):

poll = get\_object\_or\_404(Poll, id=poll\_id)

options = poll.options.all()

user\_has\_voted = request.user.is\_authenticated and Vote.objects.filter(poll=poll, user=request.user).exists()

if user\_has\_voted:

return redirect('poll\_results', poll\_id=poll.id)

if request.method == 'POST':

form = VoteForm(request.POST)

form.fields['choice'].choices = [(o.id, o.option\_text) for o in options]

if form.is\_valid():

selected\_option\_id = form.cleaned\_data['choice']

selected\_option = get\_object\_or\_404(PollOption, id=selected\_option\_id)

Vote.objects.create(poll=poll, option=selected\_option, user=request.user if request.user.is\_authenticated else None)

return redirect('poll\_results', poll\_id=poll.id)

else:

form = VoteForm()

form.fields['choice'].choices = [(o.id, o.option\_text) for o in options]

return render(request, 'polls/vote.html', {'poll': poll, 'form': form, 'user\_has\_voted': user\_has\_voted})

def poll\_list\_view(request):

polls = Poll.objects.all()

return render(request, 'polls/poll\_list.html', {'polls': polls})

def poll\_results\_view(request, poll\_id):

poll = get\_object\_or\_404(Poll, id=poll\_id)

options = poll.options.all()

option\_votes = {option: option.votes.count() for option in options}

total\_votes = sum(option\_votes.values())

option\_percentages = {option: (votes / total\_votes \* 100) if total\_votes > 0 else 0 for option, votes in option\_votes.items()}

labels = [option.option\_text for option in options]

data = [option\_votes[option] for option in options]

return render(request, 'polls/results.html', {

'poll': poll,

'option\_votes': option\_votes,

'total\_votes': total\_votes,

'option\_percentages': option\_percentages, # Pass the calculated percentages

'labels': labels,

'data': data,

})

@user\_passes\_test(lambda u: u.is\_superuser)

def analytics\_dashboard\_view(request):

# Total number of polls

total\_polls = Poll.objects.count()

# Total number of votes

total\_votes = Vote.objects.count()

# Top 5 polls with the most votes

top\_polls = Poll.objects.annotate(vote\_count=Count('votes')).order\_by('-vote\_count')[:5]

# Top 5 users who have cast the most votes

top\_voters = User.objects.annotate(vote\_count=Count('vote')).order\_by('-vote\_count')[:5]

context = {

'total\_polls': total\_polls,

'total\_votes': total\_votes,

'top\_polls': top\_polls,

'top\_voters': top\_voters,

}

return render(request, 'polls/analytics\_dashboard.html', context)

def poll\_search(request):

query = request.GET.get('query')

polls = Poll.objects.filter(title\_\_icontains=query) | Poll.objects.filter(question\_\_icontains=query)

return render(request, 'polls/poll\_list.html', {'polls': polls})

@login\_required

def user\_dashboard\_view(request):

user = request.user

polls = Poll.objects.filter(creator=user)

# Total Polls and Votes

total\_polls = polls.count()

total\_votes = Vote.objects.filter(poll\_\_creator=user).count()

poll\_data = []

for poll in polls:

options = poll.options.all()

labels = [option.option\_text for option in options]

data = [option.votes.count() for option in options]

total\_votes\_for\_poll = sum(data) # Sum votes for this poll

vote\_percentage = (total\_votes\_for\_poll / total\_votes \* 100) if total\_votes > 0 else 0

poll\_data.append({

'poll': poll,

'options': options,

'labels': labels,

'data': data,

'total\_votes': total\_votes\_for\_poll,

'vote\_percentage': vote\_percentage,

})

# Votes over time

votes\_over\_time = (Vote.objects

.filter(poll\_\_creator=user)

.annotate(date=TruncDate('timestamp')) # Assuming you have a 'timestamp' field in Vote model

.values('date')

.annotate(vote\_count=Count('id'))

.order\_by('date'))

vote\_time\_labels = [v['date'].strftime('%Y-%m-%d') for v in votes\_over\_time]

vote\_time\_data = [v['vote\_count'] for v in votes\_over\_time]

context = {

'poll\_data': poll\_data,

'total\_polls': total\_polls,

'total\_votes': total\_votes,

'vote\_time\_labels': vote\_time\_labels,

'vote\_time\_data': vote\_time\_data,

}

return render(request, 'polls/user\_dashboard.html', context)

Results

Test Case

**Integration Test**

Integration Testing is also called as “Component Testing “because it is about testing whether the system works properly when different components are integrated to form a complete system

Integration Testing is defined as set of interactions among components.Testing the interaction between modules and interaction with other systems externally is called “Integration Testing”.

INTEGRATION TEST

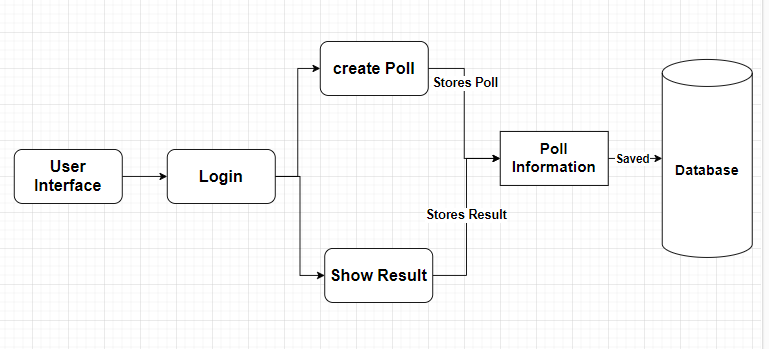
Module A

Module B

Module C

In My Project have many module ,so that I test only one module due to least time .

**Poll InformationTest**

****

**BlackBox Test**

Black Box Testing involves only the observation of the output for certain input values,and there is not attempt to analyze the programming code that enables to arrive at those outputs.Therefore,it is also termed “data-driven,input/output driven or requirements based testing”.

Black box testing is also called as”Behavioraltesting”because it only checks the system for certain inputs and doesn’t think about the internal coding.It is concerned with testing the functions of the system.Therefor,it is also called as “Functional Testing”.

No view of the code or any other internal parts system

Input Test Data

Output Data test

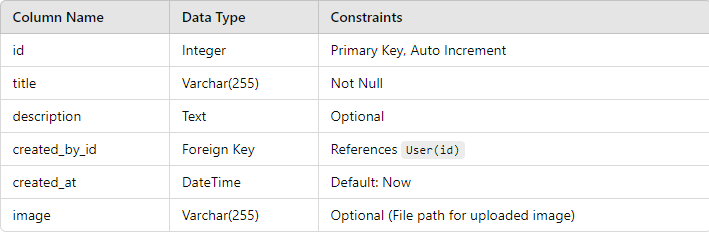
**DataBase**

**Structure**

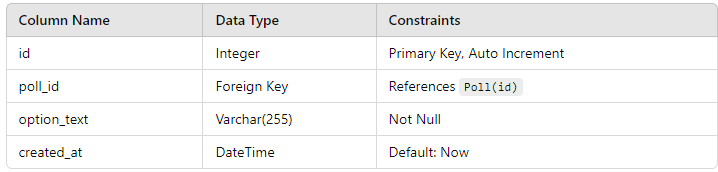
**Table Name: User Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| id | Integer | Primary Key, Auto Increment |
| username | Varchar(150) | Unique, Not Null |
| email | Varchar(254) | Unique, Not Null |
| password | Varchar(128) | Not Null |
| is\_staff | Boolean | Default: False |
| is\_superuser | Boolean | Default: False |
| date\_joined | DateTime | Not Null |

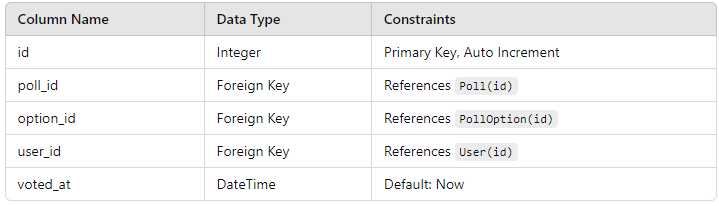
**Table Name: Poll** **Table**



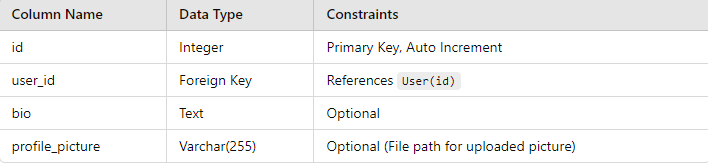
**Table Name: Poll Option Table**

****

**Table Name: Vote Table**

****

**Table Name: User Profile Table**

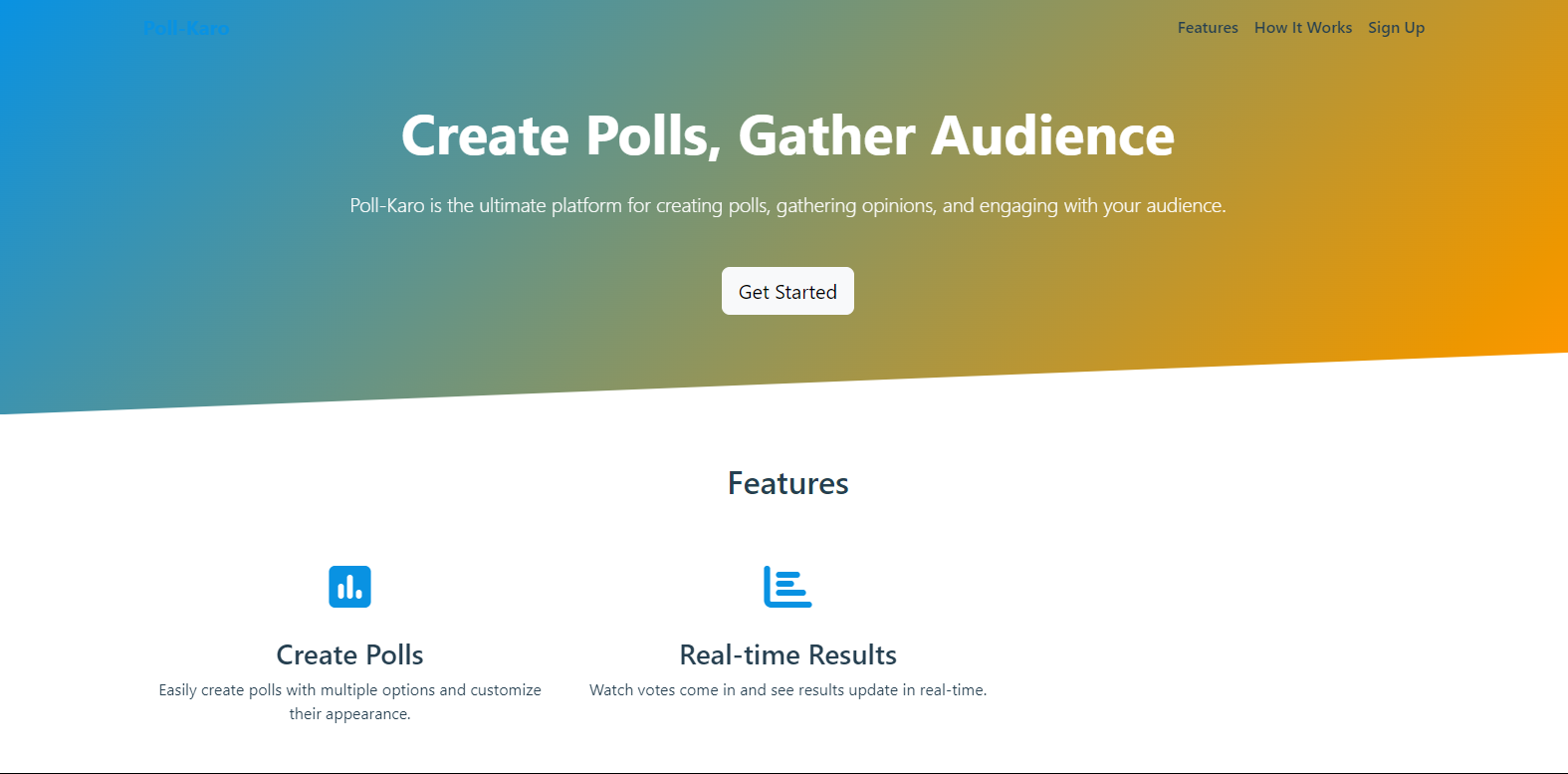
****

### ****Relationships between Tables****

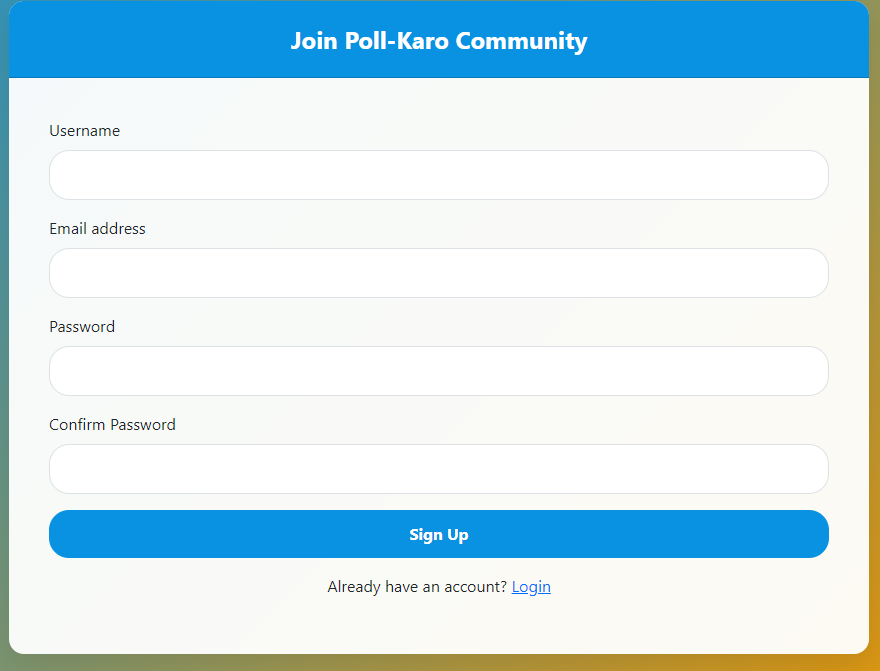
1. **User - Poll**: One user can create many polls, but a poll is created by one user. This is a **one-to-many** relationship (User → Poll).
2. **Poll - PollOption**: Each poll has multiple options. This is a **one-to-many** relationship (Poll → PollOption).
3. **PollOption - Vote**: Each option can receive many votes, but a vote is cast for one option. This is a **one-to-many** relationship (PollOption → Vote).
4. **User - Vote**: Each user can vote on many polls, but can only vote once per poll. This implies a **many-to-one** relationship (Vote → User).

**Screenshot**

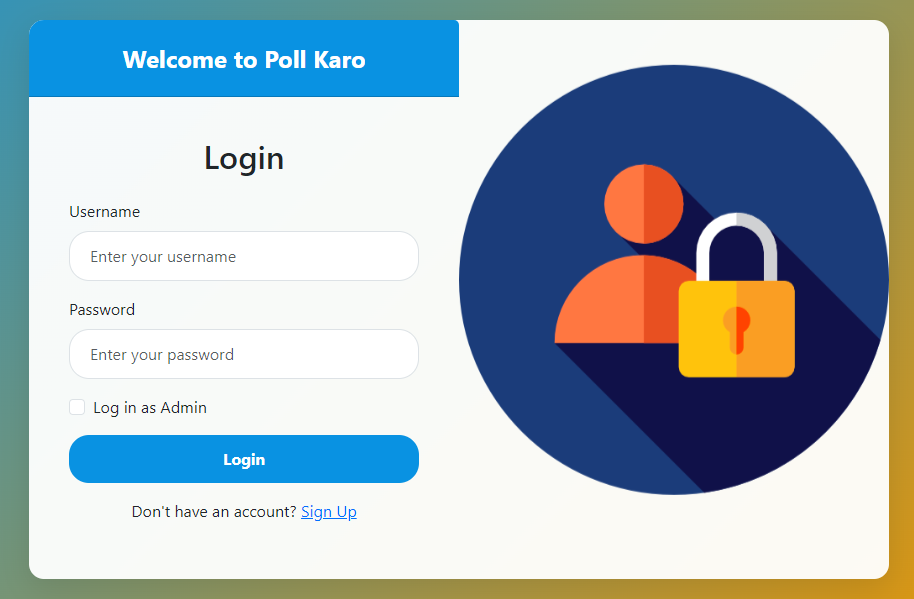
Main Page



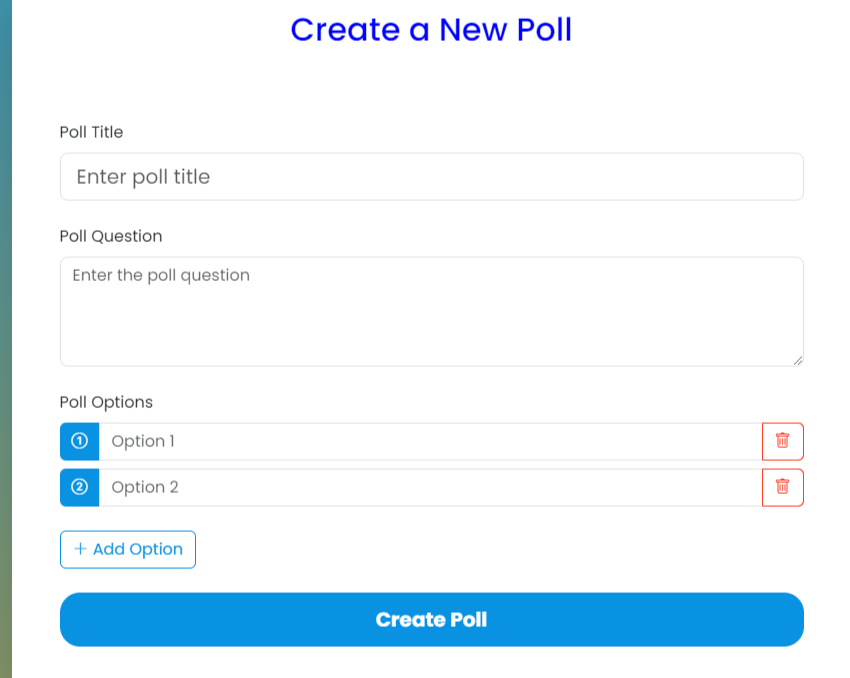
User Signup Page



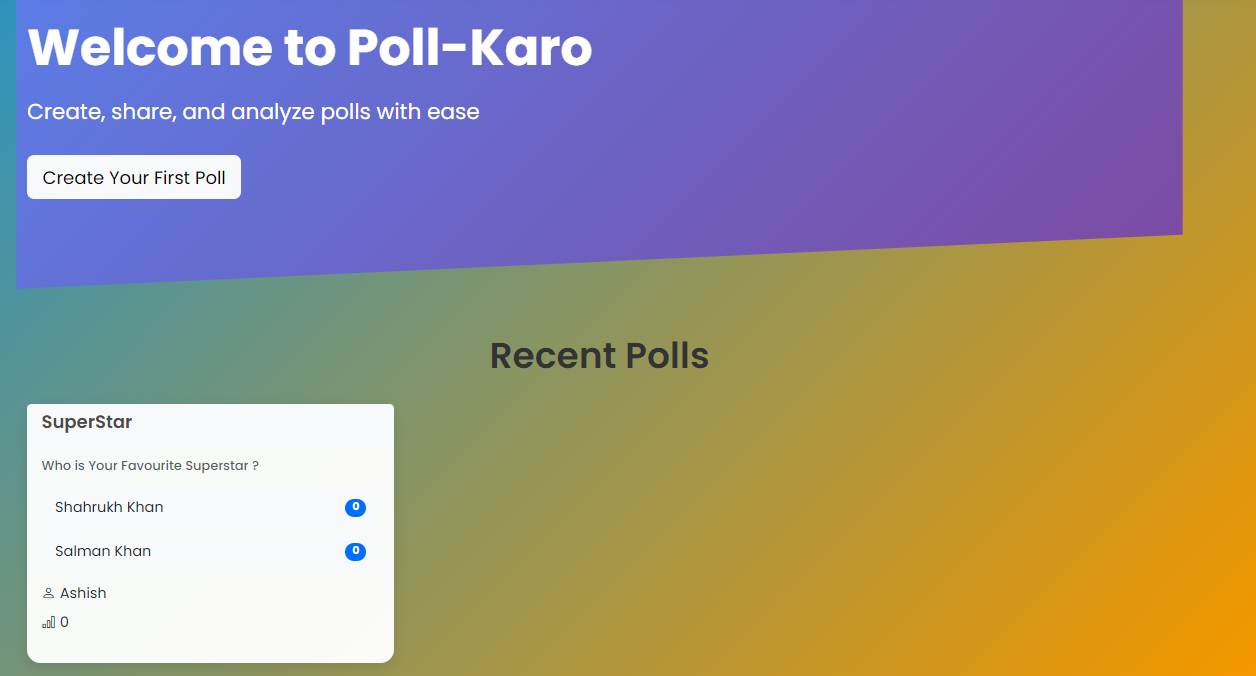
Login Page



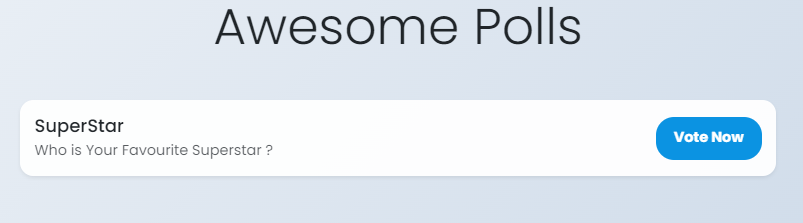
Create Poll Page



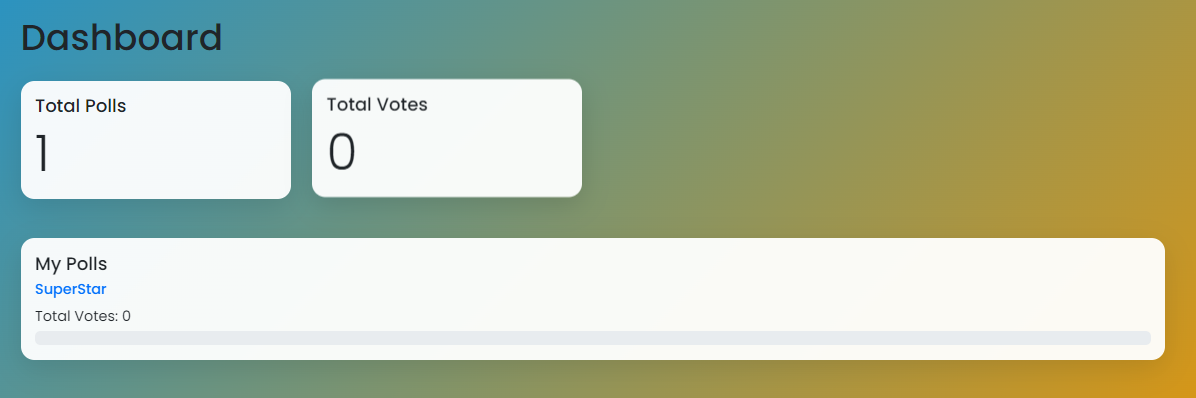
HomePage



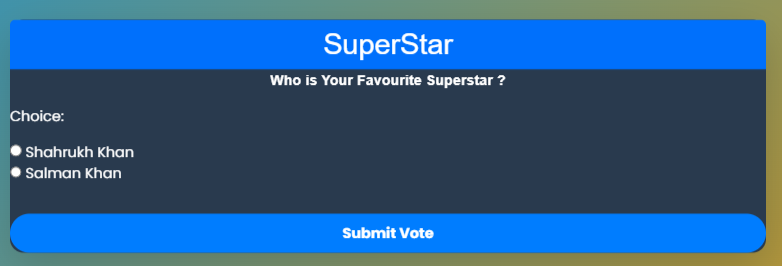
Poll List Page



User DashBoard Page



Vote Page

****

**FutureScope**

**Future Scope**

The future scope of this project includes several potential enhancements to improve functionality, scalability, and user experience. Real-time poll updates using WebSockets can enhance live interactions, while mobile app development will expand accessibility across devices.

Integration with social media platforms will boost engagement by enabling poll sharing and social login options. Adding AI-driven poll suggestions and advanced analytics will offer users deeper insights and personalized recommendations. Gamification elements, such as badges and leaderboards, can increase user participation, while multi-language support will make the platform more accessible to a global audience. Enhanced security measures, like two-factor authentication and encryption, will protect user data, and cloud integration with platforms like AWS or Google Cloud will ensure scalability as the user base grows.

Additionally, integrating blockchain technology can ensure poll transparency and security, making the platform more trustworthy. Finally, monetization through premium features and third-party API integrations can open new business opportunities and enhance user functionality.

**CONCLUSION**

**Conclusion**

In conclusion, this project successfully delivers a comprehensive and user-friendly platform for creating, managing, and visualizing polls.

By combining intuitive design with robust functionality, the application caters to a wide range of users, allowing them to engage in interactive polling experiences. The project integrates essential features like user authentication, dynamic chart displays, and real-time results, providing a seamless and efficient experience. With a scalable architecture and scope for future enhancements, this platform is poised to evolve and meet the demands of diverse users.

The project's potential for further development ensures its relevance and usability in various domains, from personal polls to large-scale surveys.

**REFERENCES**

**References:**

I referred to the following books and google for completion of my project:

**INTERNET REFERENCES:**

Reference taken from Websites like

**Django :**

**Official Django documentation was a primary resource for understanding various Django features and best practices for web development.  
URL:** [**https://docs.djangoproject.com**](https://docs.djangoproject.com)

**Stack Overflow**:

**Stack Overflow served as a valuable resource for troubleshooting and problem-solving during development.  
URL:** [**https://stackoverflow.com**](https://stackoverflow.com)