Political Party Management Application - Implementation Plan

Project Overview

A comprehensive political party management platform inspired by orgo.space, enhanced with data science capabilities for voter analytics, sentiment analysis, and member engagement optimization.

Tech Stack

Layer	Technology
Frontend	React 18 + TypeScript + Vite
Backend	Python 3.11 + FastAPI
Primary DB	PostgreSQL 15 + PostGIS (core data, hierarchy)
Analytics DB	MongoDB 6 (logs, time-series, analytics)
Cache/Queue	Redis 7 + Celery
ML Pipeline	Airflow + MLflow + scikit-learn
Deployment	Docker + AWS (ECS/EKS)

Target Scale

- Regional/State-level political party
- Hierarchy: State → District → Constituency → Booth/Ward
- Thousands of members with multi-language support

Project Structure

```
vck/
├── frontend/                      # React TypeScript Frontend
│   ├── src/
│   │   ├── components/           # Shared UI components
│   │   ├── features/             # Feature-based modules
│   │   │   ├── auth/             # Authentication
│   │   │   ├── members/          # Member management
│   │   │   ├── hierarchy/        # Org structure
│   │   │   ├── events/           # Events & campaigns
│   │   │   ├── communications/   # Messaging, announcements
│   │   │   ├── analytics/        # Dashboards
│   │   │   └── voting/           # eVoting
│   │   ├── hooks/                # Custom React hooks
│   │   ├── services/             # API clients
│   │   ├── store/                # Zustand state management
│   │   ├── i18n/               # Internationalization
│   │   └── utils/              # Utilities
│   ├── Dockerfile
│   └── package.json
│
├── backend/                      # FastAPI Backend
│   ├── src/
│   │   ├── core/               # Config, security, database
│   │   ├── auth/               # Authentication module
│   │   ├── members/            # Member management
│   │   ├── hierarchy/          # Organizational units
│   │   ├── events/             # Events & campaigns
│   │   ├── communications/     # Messaging system
│   │   ├── voting/             # eVoting system
│   │   ├── donations/          # Fund management
│   │   ├── grievances/         # Complaint handling
│   │   └── workers/            # Celery tasks
│   ├── alembic/                # Database migrations
│   ├── tests/
│   ├── Dockerfile
│   └── requirements.txt
│
├── ml-services/                  # Data Science Services
│   ├── src/
│   │   ├── voter_analytics/    # Voter prediction models
│   │   ├── sentiment/          # Sentiment analysis
│   │   ├── engagement/         # Engagement scoring
│   │   └── api/                # ML API endpoints
│   ├── models/                 # Trained model storage
│   ├── notebooks/              # Jupyter notebooks
│   └── Dockerfile
│
├── airflow/                      # ML Pipeline Orchestration
│   └── dags/
│
├── docker-compose.yml
├── docker-compose.prod.yml
└── docs/
```

Core Modules Implementation

1. Member & Hierarchy Management

Features:

- Member registration with Phone OTP, Email, Social Login
- Profile management (personal, political history, skills)
- Party hierarchy using PostgreSQL `ltree` extension
- Role/position assignments at each level
- Member transfers between units
- Digital membership card generation

Key Database Tables:

```
-- Organization Units (ltree for hierarchy)
organization_units: id, name, type (state/district/constituency/booth),
                   path (ltree), parent_id, geo_boundary (PostGIS)

-- Members
members: id, phone, email, name, photo_url, date_of_birth, gender,
         address, geo_location, status, joined_at

-- Member-Unit Relationships
member_units: member_id, unit_id, role, position, is_primary, joined_at

-- Positions
positions: id, name, level, permissions (JSONB)
```

2. Events & Campaigns

Features:

- Rally/meeting creation with geo-location
- Campaign management with goals and tracking
- Volunteer task assignment and coordination
- Attendance tracking with geo-verification
- Event analytics and reports

Key Tables:

```
events: id, title, type, unit_id, location, start_time, end_time,
        max_attendees, status, geo_point (PostGIS)

campaigns: id, name, description, unit_id, start_date, end_date,
           goals (JSONB), status

event_attendees: event_id, member_id, status, check_in_time,
                 check_in_location (PostGIS)
```

3. Communication & Engagement

Features:

- Announcements with push/SMS/email notifications
- Discussion forums with moderation
- Grievance/complaint ticketing system
- Direct messaging between members

Key Tables:

```
announcements: id, title, content, unit_id, target_scope,
               channels (push/sms/email), created_by

discussions: id, unit_id, title, content, author_id,
             is_pinned, status

grievances: id, member_id, unit_id, category, description,
            status, assigned_to, priority
```

Data Science Modules

A. Voter Analytics & Prediction

Capabilities:

- Voter database import and management
- Demographic segmentation (age, gender, location)
- Historical voting pattern analysis
- Swing voter identification using clustering
- Election outcome prediction (Random Forest, XGBoost)
- Booth-level probability mapping

MongoDB Collections:

```
voter_profiles: { voter_id, demographics, voting_history[],
                  predicted_affinity, confidence_score }

election_predictions: { election_id, booth_id, predictions[],
                        model_version, created_at }
```

B. Sentiment Analysis

Capabilities:

- Social media data collection (Twitter/X, Facebook, News)
- Real-time sentiment classification (positive/negative/neutral)
- Topic extraction and trending issues
- Competitor party monitoring
- Alert system for sentiment spikes
- Regional language support (Hindi, Tamil, Telugu, etc.)

MongoDB Collections:

```
social_posts: { source, content, sentiment_score, topics[],
                location, timestamp }
```

```
sentiment_aggregates: { date, region, topic, avg_sentiment,
                       volume, trend }
```

C. Member Engagement Scoring

Capabilities:

- Activity tracking (event attendance, app usage, donations)
- Engagement score calculation (0-100)
- Churn prediction model
- Personalized re-engagement recommendations
- Gamification (badges, leaderboards, rewards)

Key Tables:

```
member_activities: member_id, activity_type, points, timestamp

engagement_scores: member_id, score, components (JSONB),
                  churn_probability, last_calculated

badges: id, name, description, criteria (JSONB), icon_url

member_badges: member_id, badge_id, earned_at
```

Authentication Flow

```
┌──────────────────────────────────────────────────────────────┐
│                    AUTHENTICATION OPTIONS                     │
├──────────────────────────────────────────────────────────────┤
│                                                               │
│  1. Phone OTP (Primary)                                       │
│     User → Enter Phone → Request OTP → Verify OTP → JWT      │
│                                                               │
│  2. Email + Password                                          │
│     User → Enter Email/Password → Verify → JWT               │
│                                                               │
│  3. Social Login (Google/Facebook)                           │
│     User → OAuth Flow → Callback → Create/Link Account → JWT│
│                                                               │
├──────────────────────────────────────────────────────────────┤
│  JWT Structure:                                               │
│  - Access Token: 15 min expiry                               │
│  - Refresh Token: 7 day expiry (stored in Redis)             │
│  - Permissions: role-based + unit-scoped                     │
└──────────────────────────────────────────────────────────────┘
```

API Structure

```
/api/v1/
├── /auth
│   ├── POST /register          # New member registration
│   ├── POST /login/phone       # Phone OTP login
│   ├── POST /login/email       # Email password login
│   ├── POST /login/social      # OAuth callback
│   ├── POST /verify-otp        # OTP verification
│   ├── POST /refresh           # Token refresh
│   └── POST /logout            # Logout
│
├── /members
│   ├── GET /                   # List members (paginated, filtered)
│   ├── GET /{id}               # Get member details
│   ├── PUT /{id}               # Update member
│   ├── POST /{id}/transfer     # Transfer to another unit
│   └── GET /{id}/activities    # Member activity history
│
├── /hierarchy
│   ├── GET /units              # List all units (tree structure)
│   ├── GET /units/{id}         # Get unit details
│   ├── POST /units             # Create unit
│   ├── PUT /units/{id}         # Update unit
│   └── GET /units/{id}/members # Members in unit
│
├── /events
│   ├── GET /                   # List events
│   ├── POST /                  # Create event
│   ├── GET /{id}               # Event details
│   ├── POST /{id}/register     # Register for event
│   ├── POST /{id}/check-in     # Check-in with geo
│   └── GET /{id}/analytics     # Event analytics
│
├── /campaigns
│   ├── GET /                   # List campaigns
│   ├── POST /                  # Create campaign
│   └── GET /{id}/progress      # Campaign progress
│
├── /communications
│   ├── POST /announcements     # Create announcement
│   ├── GET /discussions        # List discussions
│   ├── POST /discussions       # Create discussion
│   └── GET /grievances         # List grievances
│
├── /analytics                  # Data Science APIs
│   ├── GET /voter/demographics # Voter demographics
│   ├── GET /voter/predictions  # Election predictions
│   ├── GET /sentiment/current  # Current sentiment
│   ├── GET /sentiment/trends   # Sentiment trends
│   ├── GET /engagement/scores  # Member engagement
│   └── GET /engagement/at-risk # Churn risk members
```

```
  │
  └── /voting
      ├── GET /elections          # List internal elections
      ├── POST /elections         # Create election
      └── POST /elections/{id}/vote # Cast vote
```

Third-Party Integrations

Service	Provider Options	Purpose
SMS OTP	MSG91, Twilio, AWS SNS	Phone authentication
Push Notifications	Firebase FCM, OneSignal	Mobile notifications
Email	SendGrid, AWS SES	Email communications
Social Login	Google OAuth, Facebook Login	Alternative auth
Maps	Google Maps, Mapbox	Geo-location features
Social Data	Twitter API, Meta API	Sentiment analysis
Payments	Razorpay, Stripe	Donation processing
Storage	AWS S3, MinIO	File storage

Implementation Phases

Phase 1: Foundation

- [ ] Project setup (monorepo, Docker, CI/CD)
- [ ] Database schema design and migrations
- [ ] Authentication system (Phone OTP, Email, Social)
- [ ] Basic member CRUD operations
- [ ] Organization hierarchy with ltree

Phase 2: Core Features

- [ ] Member management UI
- [ ] Hierarchy visualization and management
- [ ] Role and permission system
- [ ] Events module (CRUD, registration, attendance)
- [ ] Campaign management

Phase 3: Communication

- [ ] Announcements with multi-channel delivery
- [ ] Discussion forums
- [ ] Grievance handling system
- [ ] Notification preferences

Phase 4: Data Science - Analytics

- [ ] Voter database management
- [ ] Demographic analysis dashboards
- [ ] Member engagement scoring
- [ ] Activity tracking and gamification

Phase 5: Data Science - ML Models

- [ ] Sentiment analysis pipeline
- [ ] Voter prediction models
- [ ] Churn prediction
- [ ] ML model serving infrastructure

Phase 6: Advanced Features

- [ ] eVoting system
- [ ] Donation management
- [ ] Document management
- [ ] Advanced reporting

Phase 7: Polish & Launch

- [ ] Multi-language support (i18n)
- [ ] Performance optimization
- [ ] Security audit
- [ ] Production deployment
- [ ] User training materials

Key Files to Create First

1. `backend/src/core/config.py` - Application configuration
2. `backend/src/core/database.py` - PostgreSQL + MongoDB connections
3. `backend/src/auth/service.py` - Authentication logic
4. `backend/src/hierarchy/models.py` - Organization models with ltree
5. `backend/src/members/models.py` - Member data models
6. `frontend/src/features/auth/LoginForm.tsx` - Primary entry point
7. `docker-compose.yml` - Development environment
8. `alembic/versions/001_initial.py` - Initial DB migration

Notes

- Start with Phone OTP as primary auth (most common in India)
- Use PostGIS for booth boundary mapping and geo-verification
- MongoDB time-series collections for efficient sentiment data
- Celery for async tasks (SMS, notifications, ML jobs)
- Airflow for scheduled ML pipeline runs

- i18next for frontend internationalization