Online Poll System Backend

🎯 Overview

A production-ready **Django REST Framework** voting platform with secure JWT authentication, role-based access control, and real-time result computation. Built for scalability and easy frontend integration.

✨ Key Features

🔐 Authentication & Authorization

- **Email-based authentication** with custom user model
- **JWT token management** (access & refresh tokens)
- **Role-based access control** (Admin & Voter roles)
- Token blacklisting for secure logout

📊 Poll Management

- Create polls with multiple options and expiry dates
- Full CRUD operations for poll management
- Automatic poll expiry handling
- Prevention of modifications after expiry

🗳️ Voting System

- One vote per user per poll enforcement
- Duplicate vote prevention with database constraints
- Vote blocking after poll expiry
- Concurrent voting protection

📈 Results & Analytics

- Real-time vote tallying with caching
- Optimized PostgreSQL queries for performance
- Cache invalidation on new votes
- Vote count consistency checks

🛡️ Security Features

- SQL injection prevention
- XSS protection in API responses
- Rate limiting on voting endpoints
- CSRF and secure cookie configuration

📖 **API Documentation**

- Interactive **Swagger UI** at `/auth/docs/`
- OpenAPI schema (JSON/YAML formats)
- Complete endpoint documentation

🛠️ Tech Stack

Technology	Purpose
Django 5.2	Web framework
Django REST Framework	RESTful API development
PostgreSQL	Production database
SQLite	Development/testing database
SimpleJWT	JWT authentication
drf-yasg	API documentation
pytest	Testing framework
Gunicorn	WSGI server
WhiteNoise	Static file serving
CORS Headers	Cross-origin support

🚀 Quick Start

Prerequisites

- Python 3.12+
- PostgreSQL (for production)
- pip & virtualenv

Installation

1. **Clone the repository**

bash

```
git clone <repository_url>
cd Online_Poll_System
```

2. **Create virtual environment**

bash

```
python -m venv polls_venv

# Linux/macOS/WSL2
source polls_venv/bin/activate

# Windows
polls_venv\Scripts\activate
```

3. **Install dependencies**

bash

```
pip install -r requirements.txt
```

4. **Set up environment variables**

bash

```
# Create .env file in project root
cp .env.example .env

# Edit .env with your settings
SECRET_KEY=your-secret-key-here
DEBUG=True
DJANGO_ENV=development
DATABASE_URL=postgresql://user:pass@localhost/dbname  # Optional for dev
```

5. **Run migrations**

bash

```
python manage.py migrate
```

6. **Create superuser (optional)**

```bash
python manage.py createsuperuser
```

7. **Run development server**

```bash
python manage.py runserver
```

8. **Access the application**
   - API Base: `http://127.0.0.1:8000/`
   - Swagger Docs: `http://127.0.0.1:8000/auth/docs/`
   - Django Admin: `http://127.0.0.1:8000/admin/`

📡 API Endpoints

Authentication (`/auth/`)

Method	Endpoint	Description	Auth Required
POST	`/auth/register/`	Register new voter	❌
POST	`/auth/login/`	Login & get JWT tokens	❌
POST	`/auth/refresh/`	Refresh access token	❌
POST	`/auth/logout/`	Logout (blacklist token)	✅
GET	`/auth/me/`	Get current user profile	✅
GET	`/auth/users/`	List all users	✅ Admin
POST	`/auth/create_admin/`	Create admin user	✅ Admin
GET	`/auth/docs/`	Swagger API documentation	❌

Polls (`/api/polls/`)

Method	Endpoint	Description	Auth Required
GET	`/api/polls/`	List all polls	❌
POST	`/api/polls/`	Create new poll	✅ Admin
GET	`/api/polls/{id}/`	Get poll details	❌
PUT/PATCH	`/api/polls/{id}/`	Update poll	✅ Admin/Owner
DELETE	`/api/polls/{id}/`	Delete poll	✅ Admin/Owner
POST	`/api/polls/{id}/vote/`	Cast a vote	✅
GET	`/api/polls/{id}/results/`	Get poll results	❌

💡 Usage Examples

Register a New User

```bash
POST /auth/register/
Content-Type: application/json

{
  "first_name": "John",
  "surname": "Doe",
  "email": "john@example.com",
  "confirm_email": "john@example.com",
  "password": "SecurePass123",
  "confirm_password": "SecurePass123"
}
```

Login & Get Tokens

```bash
POST /auth/login/
Content-Type: application/json

{
  "email": "john@example.com",
  "password": "SecurePass123"
}

# Response
{
  "access": "eyJ0eXAiOiJKV1QiLCJhbGc...",
  "refresh": "eyJ0eXAiOiJKV1QiLCJhbGc..."
}
```

Create a Poll (Admin)

```bash
POST /api/polls/
Authorization: Bearer <access_token>
Content-Type: application/json

{
  "question": "What's your favorite programming language?",
  "options": [
    {"text": "Python"},
    {"text": "JavaScript"},
    {"text": "Go"}
  ],
  "expires_at": "2025-12-31T23:59:59Z"
}
```

Vote on a Poll

```bash
POST /api/polls/{poll_id}/vote/
Authorization: Bearer <access_token>
Content-Type: application/json

{
  "option_id": 1
}
```

Get Poll Results

```bash
GET /api/polls/{poll_id}/results/
```

🧪 Testing

Run the complete test suite:

```bash
# Run all tests
pytest -v

# Run with coverage
pytest --cov=api --cov=polls

# Run specific test file
pytest api/test/test_api_unit.py -v

# Run specific test
pytest api/test/test_api_unit.py::test_register_voter -v
```

Test Coverage:

- ✅ 45 tests passing
- ✅ Authentication & authorization
- ✅ Poll CRUD operations
- ✅ Voting logic & constraints
- ✅ Security (SQL injection, XSS, rate limiting)
- ✅ Cache invalidation
- ✅ Edge cases & error handling

🚢 Deployment

Environment Variables

Create a `.env` file with:

```env
# Security
SECRET_KEY=your-production-secret-key
DEBUG=False
ALLOWED_HOSTS=yourdomain.com,www.yourdomain.com

# Database
DATABASE_URL=postgresql://user:password@host:5432/database

# Email (optional)
EMAIL_HOST=smtp.gmail.com
EMAIL_PORT=587
EMAIL_HOST_USER=your-email@gmail.com
EMAIL_HOST_PASSWORD=your-app-password

# CORS
CORS_ALLOWED_ORIGINS=https://yourdomain.com

# SSL/Security
SECURE_SSL_REDIRECT=True
```

Production Checklist

- [ ] Set `DEBUG=False`
- [ ] Configure `SECRET_KEY`
- [ ] Set up PostgreSQL database
- [ ] Configure `ALLOWED_HOSTS`
- [ ] Set up static files (`collectstatic`)
- [ ] Configure CORS origins
- [ ] Enable SSL/HTTPS
- [ ] Set up email backend
- [ ] Configure caching (Redis recommended)
- [ ] Run migrations
- [ ] Create superuser

Deployment Commands

```bash
# Collect static files
python manage.py collectstatic --noinput

# Run migrations
python manage.py migrate

# Start with Gunicorn
gunicorn online_poll_system.wsgi:application --bind 0.0.0.0:8000
```

📁 Project Structure

```
Online_Poll_System/
├── api/                    # Authentication & user management
│   ├── models.py           # Custom User model
│   ├── serializers.py      # API serializers
│   ├── views.py            # Authentication views
│   ├── permissions.py      # Custom permissions
│   ├── urls.py             # Auth endpoints
│   └── test/               # Authentication tests
├── polls/                  # Poll management
│   ├── models.py           # Poll, Option, Vote models
│   ├── serializers.py      # Poll serializers
│   ├── views.py            # Poll viewsets
│   ├── urls.py             # Poll endpoints
│   └── tests/              # Poll tests
├── online_poll_system/     # Project settings
│   ├── settings.py         # Django settings
│   ├── urls.py             # Root URL config
│   └── wsgi.py             # WSGI config
├── requirements.txt        # Python dependencies
├── pytest.ini              # Pytest configuration
├── .env                    # Environment variables (not in repo)
└── manage.py               # Django management script
```

🤝 Contributing

Contributions are welcome! Please:

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/AmazingFeature`)
3. Commit your changes (`git commit -m 'Add some AmazingFeature'`)
4. Push to the branch (`git push origin feature/AmazingFeature`)
5. Open a Pull Request

📄 License

👨‍💻 Author

Akindipe Muheez Omogbolahan

- 📧 Email: akindipemuheez@outlook.com
- 🔗 LinkedIn: akinscoded
- 💻 GitHub: Akins-Coded
- 🌐 Website: akinscoded.kit.com
- 🌳 Linktree: akinscoded

🙏 Acknowledgments

- Django REST Framework community
- Contributors and testers
- Open source libraries used in this project

<div align="center">

Built with precision, security, and scalability in mind 🚀

⭐ Star this repo if you find it helpful!

</div>