

# Project Technical Report: The Districts Projects

## 1. Overview

The Districts Projects is a content-heavy data visualization and management platform. The system is architected as a modular monolith using the Django framework. It prioritizes server-side rendering for SEO and initial load performance, utilizing Tailwind CSS for a modern, responsive UI.

The application is designed to handle complex relationships between cultural data, statistical metrics, and geographical entities (States/Districts), supported by a robust MySQL database backend.

## 2. Technology Stack

### Backend Core-

- **Language:** Python 3.13.5
- **Framework:** Django 5.2.3
- **Server Gateway:** WSGI (Gunicorn in production)
- **Environment:** python-dotenv for secure secret key and credential management.

### Frontend & UI

- **Templating:** Django Template Language (DTL) – Server-side rendering.
- **Styling:** Tailwind CSS (integrated via django-tailwind with JIT compilation).
- **Content Editing:** TinyMCE 4 (Rich Text Editor) for advanced content creation in the admin panel.
- **Data Visualization:** Chart.js Used for rendering interactive statistical graphs (bar, line charts) within the statistic module.

### Data & Storage

- **Database:** MySQL (Production & Dev). Configured with STRICT\_TRANS\_TABLES for data integrity.
- **Media Storage:** Local filesystem storage, served via Nginx.
- **Static Assets:** Managed via WhiteNoise (CompressedManifestStaticFilesStorage) for efficient caching and serving.

### Key Utilities

- **Image Processing:** django-imagekit for automated resizing.
- **Polymorphism:** django-polymorphic to handle complex content inheritance (e.g., different types of cultural entries sharing a base model).
- **Data Import:** Custom importdata app for bulk processing of datasets.

### 3. System Architecture

The project follows the standard Model-View-Template (MVT) architectural pattern. the logic is decoupled into specific functional Apps.

**Instead of a single large application, the codebase is split into domain-specific modules:**

**Core & Navigation:** home, search, footersection, sidepanal.

**Logic & Data:**

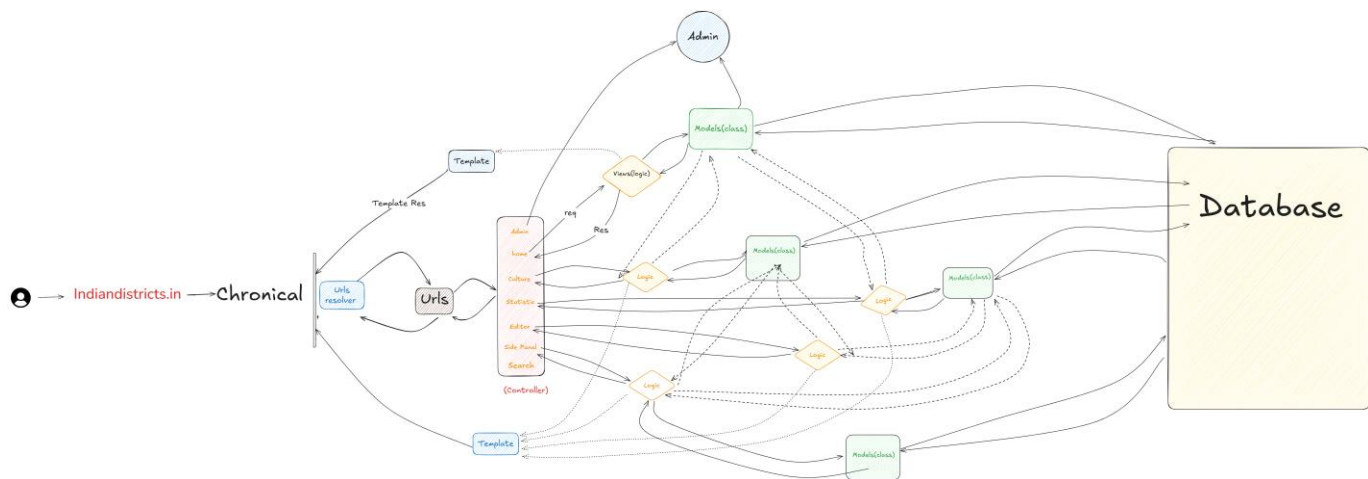
- **culture:** Manages festivals, traditions, and cultural artifacts.
- **statistic:** Handles numerical data points and visualization logic.
- **users:** Custom authentication and user profile management.

**Administration:**

- **admindashboard:** A custom high-level view for platform administrators.
- **editor:** Specialized interfaces for content editor.

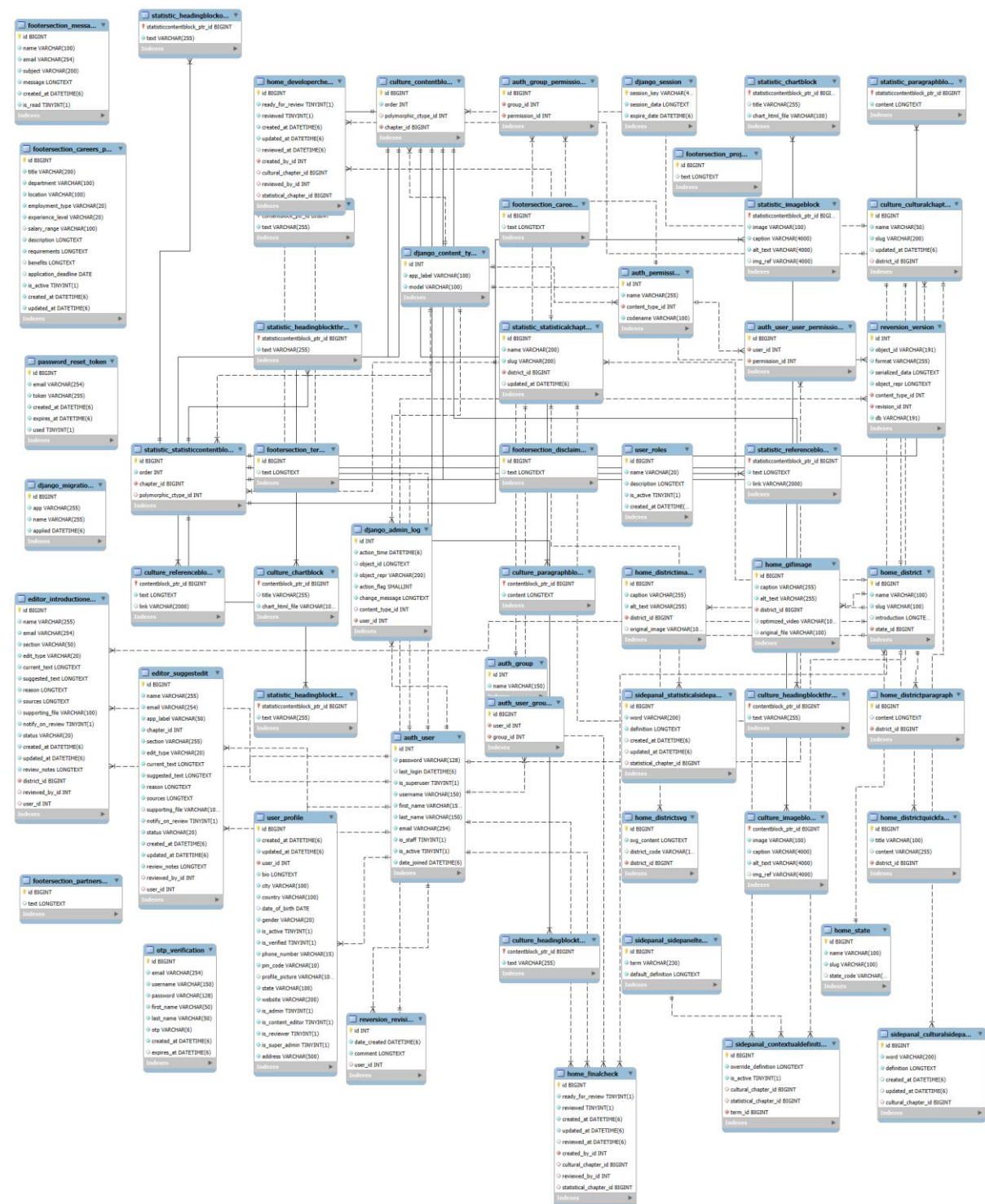
#### Architectural Diagram

The following diagram illustrates the request flow from the client, through the Nginx reverse proxy, into the Django application layers, and finally to the MySQL database.



## 4. Database Design

The database is MySQL & We are utilizing Django's ORM to abstract raw SQL



## 5. Deployment Infrastructure

The production environment is hosted on a VPS.

1. Server - Hostinger VPS
2. OS: Linux (Ubuntu)
3. CPU: 4 vCPU
4. RAM: 16 GB
5. Storage: 200 GB NVMe SSD
6. Bandwidth: 16TB

### Production Pipeline

#### 1.Nginx (Reverse Proxy Web Server):

- Acts as the entry point for all traffic.
- Handles SSL Termination (HTTPS via Let's Encrypt).
- Serves Media files
- Forwards requests to Gunicorn socket.

#### 2.Gunicorn (Application Server):

- Runs the Python/Django code.
- Configured with **9 workers** to handle concurrent requests efficiently.

#### 3.Security Configuration:

- Firewall: Ports locked down to 80 (HTTP), 443 (HTTPS), and 22 (SSH).