NFC CERTIFICATION PROJECT

Digital Forest Certification & Verification Platform

NFC

Nepal Forest Certification System

TECHNICAL DOCUMENTATION

for the NFC Certification Project

Version: [1.0]

Published: [June, 2026]

Django + DRF · Next.js + TypeScript · PostgreSQL · Enketo · Nginx

Table of Contents

01 Project Overview

Architecture, features and technology stack

The NFC Certification project is a comprehensive application designed to manage and monitor various aspects of a system. It integrates a robust backend powered by Django and a modern frontend built with Next.js. The project includes features such as user authentication, data processing, applicant registration, examiner assignment, field verification, form submissions, administrative review, and approval management.

1.1 System Capabilities

The system supports the following capabilities:

- Applicant onboarding and profile management
- Examiner application and approval workflows
- Dynamic assignment management
- Verification and inspection processes
- Enketo form integration for mobile-friendly field data collection
- Administrative dashboards and analytics
- Role-based access control
- Status tracking and workflow automation

1.2 Technology Stack

The platform is built using the following technologies:

Layer	Technology
Frontend	Next.js + TypeScript + Tailwind CSS
Backend	Django + Django REST Framework
Database	PostgreSQL
Reverse Proxy	Nginx
Deployment	Linux VPS / Ubuntu Server
Form Engine	Enketo

02 Project Structure

Repository layout and directory organization

2.1 Root Directory

- client/ — Frontend application built with Next.js.
- server/ — Backend application powered by Django.
- enketo/ — Submodule for form management and processing.
- deployment/ — Deployment scripts and configurations.

2.2 Client Directory

Key folders and files:

- app/ — Contains pages and layouts for the application.
  - globals.css — Global styles.
  - layout.tsx — Application layout.
  - page.tsx — Main landing page.
  - Subfolders for specific pages: about/, admin/, login/, etc.
- components/ — Reusable UI components.
  - admin/ — Components for admin pages.
  - submission/ — Components for submission handling.
  - ui/ — General-purpose UI components.
- hooks/ — Custom React hooks.
- lib/ — Utility functions and context providers.
- public/ — Static assets like images and fonts.
- styles/ — Global CSS files.
- next.config.ts — Next.js configuration file.

2.3 Server Directory

Key folders and files:

- config/ — Configuration files for different environments.
- settings/ — Environment-specific settings.
- nfc_redd_backend/ — Core backend logic.
- manage.py — Entry point for Django commands.
- docker-compose.yml — Docker configurations for various environments.

2.4 Enketo Directory

Key folders and files:

- app/ — Contains controllers, models, and views.
- controllers/ — Handles API and business logic.
- config/ — Configuration files for Enketo.
- public/ — Static assets for Enketo.

2.5 Deployment Directory

Key folders and files:

- staging/ — Deployment configurations for the staging environment.
- nginx_nfc_staging.conf — Nginx configuration.
- start_gunicorn_staging.sh — Script to start Gunicorn.
- supervisor_bipad_mis_staging.conf — Supervisor configuration.

03 Setup Instructions

Prerequisites, backend, frontend and Docker setup

3.1 Prerequisites

- Python 3.10+
- Node.js 18+
- Docker
- PostgreSQL
- GDAL and GEOS libraries (for geospatial data processing)

3.2 Backend Setup

1. Clone the repository.
2. Create a virtual environment:

```
python -m venv venv
source venv/bin/activate
```

3. Install dependencies:

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

4. Set up environment variables:
   - Copy .env.example to .env.
   - Update values as needed.
5. Apply migrations:

```
python manage.py migrate
```

6. Run the development server:

```
python manage.py runserver
```

3.3 Frontend Setup

1. Navigate to the client directory.
2. Install dependencies:

```
npm install
```

3. Run the development server:

```
npm run dev
```

3.4 Docker Setup

1. Build and start the containers:

```
docker-compose up --build
```

04 Backend (Server)

Django framework features and configuration

4.1 Key Features

- Django Framework — Provides robust backend capabilities.
- Celery — Handles asynchronous tasks.
- Django Debug Toolbar — Debugging tool for development.
- GDAL and GEOS — Libraries for geospatial data processing.

4.2 Configuration

Configuration file: config/settings/local.py

Key settings:

- DEBUG — Enables debug mode.
- ALLOWED_HOSTS — Specifies allowed hosts.
- INSTALLED_APPS — Includes additional apps like debug_toolbar and django_extensions.

05 Frontend (Client)

Next.js framework features and configuration

5.1 Key Features

- Next.js Framework — Server-side rendering and static site generation.
- Reusable Components — Modular design with components like navbar.tsx and glass-card.tsx.
- Custom Hooks — Includes hooks like use-toast and use-mobile.

5.2 Configuration

Configuration file: next.config.ts

Key settings:

- i18n — Internationalization support.
- publicRuntimeConfig — Runtime configuration for the client.

06 Deployment

Staging environment and infrastructure components

6.1 Staging Environment

- Nginx — Handles reverse proxy and static file serving.
- Gunicorn — WSGI server for Django.
- Supervisor — Manages process control.

6.2 Key Files

- nginx_nfc_staging.conf
- start_gunicorn_staging.sh
- supervisor_bipad_mis_staging.conf

07 Enketo Integration

Mobile-friendly field data collection workflow

7.1 Purpose

Enketo is integrated to provide:

- Mobile-friendly data collection
- Offline-capable forms
- Structured field verification
- Real-time submission access

7.2 Submission Workflow

1. Assignment created
2. Applicant opens Enketo Form
3. Form submitted
4. Submission synced to backend
5. Examiner and Admin reviews submission

7.3 Verification Flow

1. Admin assigns examiner
2. Examiner receives assignment
3. Examiner opens Enketo form
4. Field verification is completed
5. Submission is stored
6. Admin reviews verification
7. Final decision is recorded

08 Security Considerations

Authentication, data protection and access control

The NFC Certification platform implements the following security features:

- JWT/token authentication
- Protected APIs
- CSRF protection
- HTTPS deployment
- Role-based permissions
- Input validation
- Secure password hashing

Important

Security configuration values such as secret keys, tokens and credentials should never be committed to version control and must be managed through environment variables.

09 Environment Variables

Centralized configuration for services and integrations

The project uses a centralized .env configuration for managing application services, credentials, integrations, and deployment settings.

9.1 General Configuration

```
USE_DOCKER=yes
IPYTHONDIR=/app/.ipython
```

9.2 Redis Configuration

```
REDIS_URL=redis://redis:6379/0
```

Redis is used for:

- Celery message broker
- Background task queue
- Caching support
- Asynchronous processing

9.3 Celery and Flower Configuration

```
CELERY_FLOWER_USER=your_flower_username
CELERY_FLOWER_PASSWORD=your_flower_password
```

Purpose:

- Celery handles asynchronous task processing.
- Flower provides monitoring and management for Celery workers.

9.4 Database Configuration

Local PostgreSQL configuration:

```
POSTGRES_HOST=postgres
POSTGRES_PORT=5432
POSTGRES_DB=nfc_redd_backend
POSTGRES_USER=postgres_user
POSTGRES_PASSWORD=postgres_password
```

Database responsibilities:

- User management
- Verification records
- Assignment tracking

- Form submission storage
- Administrative workflows

9.5 Email Configuration

```
EMAIL_HOST_USER=example@gmail.com
EMAIL_HOST_PASSWORD=your_app_password
```

The email service is used for:

- Approval notifications
- Password reset workflows
- Administrative alerts

> **Note**
> *Replace all placeholder credentials shown above with secure, environment-specific values before deploying to production.*

10 Conclusion

Summary, architecture benefits and future outlook

The NFC Certification project provides a scalable and modular architecture for managing verification workflows digitally. The platform combines modern frontend technologies with a robust backend system to support applicants, examiners, and administrators efficiently.

10.1 Architecture Benefits

The project architecture allows:

- Scalable deployment
- Secure authentication
- Structured verification workflows
- Efficient administrative management
- Future extensibility

The system is suitable for enterprise-scale environmental verification and digital governance workflows.