FlyNova Project Documentation

Project Name: FlyNova

Version: 1.0

Date: December 2025

Developer: Md. Shariful Islam Rony

Table of Contents

Page 1: Executive Summary & Project Overview

1.1 Introduction

FlyNova is a comprehensive, web-based flight booking management system designed to streamline the travel experience for users in Bangladesh and beyond. Built with simplicity and efficiency in mind, FlyNova provides a robust platform for searching domestic and international flights, managing bookings, and exploring holiday packages and hotel accommodations.

1.2 Purpose & Scope

The primary goal of FlyNova is to bridge the gap between travelers and travel services by offering a centralized, user-friendly interface.

- **For Users:** It offers a seamless way to compare flight prices, view real-time flight status, book tickets, and manage travel itineraries.
- **For Administrators:** It provides a powerful backend to manage flight schedules, update airport information, handle bookings, and curate holiday packages.

1.3 Core Value Proposition

- **Smart Search:** Advanced filtering capabilities allow users to find flights based on price (Low, Mid, High), dates, and destinations.
- **Real-Time Information:** A live airport dashboard provides up-to-date arrival and departure status modifications.
- **Holistic Travel Solution:** Beyond flights, FlyNova integrates hotel bookings and holiday packages, serving as a one-stop shop for travelers.
- **User-Centric Design:** A responsive, modern UI ensures accessibility across desktop and mobile devices.

1.4 Target Audience

- **Individual Travelers:** Looking for quick and easy flight bookings.
- **Corporate Clients:** Managing business travel with downloadable PDF tickets.
- **Travel Enthusiasts:** Exploring holiday packages and hotel deals.

Page 2: Technical Architecture & Technology Stack

2.1 System Architecture

FlyNova follows the **Model-View-Template (MVT)** architectural pattern, the standard for Django applications. This ensures a clean separation of concerns:

- **Model Layer:** Handles database structure and business logic (e.g., flight pricing, booking status).
- **View Layer:** Processes user requests, interacts with models, and determines the data to send to the user.
- **Template Layer:** Presentation layer using HTML/CSS to render dynamic data to the user.

2.2 Technology Stack

The project leverages a robust set of open-source technologies:

Backend

- **Framework: Django 5.2.8** (Python Web Framework) - Chosen for its security, scalability, and "batteries-included" approach.
- **Language: Python 3.8+** - known for readability and vast ecosystem.
- **Database: SQLite** (Development) / **PostgreSQL** (Recommended for Production) - Relational database management.

Frontend

- **HTML5 & CSS3:** For structure and styling.
- **Bootstrap 5:** For responsive layout and pre-built components.
- **JavaScript:** For dynamic client-side interactions.
- **Template Engine:** Django Template Language (DTL).

Key Libraries & Dependencies

- **django-allauth (0.57.0):** Handles authentication, registration, and account management, including Google OAuth social login.
- **ReportLab (4.0.7):** Engine for generating dynamic PDF tickets programmatically.
- **Pillow (12.0.0):** Image processing library for handling airline logos, hotel images, and package thumbnails.
- **Tzdata (2025.2):** Timezone support to ensure accurate flight scheduling across global time zones.

2.3 Directory Structure

The project is organized into modular applications for maintainability:

- `config/`: Main project settings and URL routing.
- `accounts/`: User management, custom user model, and authentication views.
- `core/`: Base templates, home page logic, and static landing page features.
- `flights/`: Flight management, airport data, and search logic.
- `bookings/`: Order processing, payment simulation, and ticket generation.
- `hotels/`: Hotel inventory and room booking system.
- `packages/`: Holiday package curation and display.

Page 3: User Roles & Feature Specifications

3.1 User Roles

FlyNova supports two primary user roles, each with distinct permissions:

1. Guest User:

- Can search for flights, hotels, and packages.
- Can view flight details and real-time airport status.
- Must register/login to make a booking.

2. Registered User:

- **Dashboard:** Access to a personal profile.
- **Booking:** Can book flights, hotels, and packages.
- **History:** View past and upcoming bookings.
- **Ticket Management:** Download PDF tickets for confirmed bookings.

3. Administrator (Superuser):

- Full access to the Django Admin Panel.
- Manage Users (Create, Delete, Update).
- Manage Content (Add Flights, Airports, Hotels, Packages).
- Manage Bookings (View details, Override status).

3.2 Core Modules & Features

A. Flight Module (`flights` app)

- **Search Engine:** Users can query by Origin, Destination, Date, and Passenger count.
- **Smart Filtering:**
- **Price Filter:** Dynamic categorization into Low, Mid, and High price ranges based on current market data.
- **Sorting:** Results defaulted to the most recent departure times.
- **Airport Dashboard:** "Live" view of flights arriving in the next 24 hours with statuses like "On Time," "Landing Soon," or "Landed."

B. Booking System (`bookings` app)

- **Unified Booking Model:** Uses Django's `GenericForeignKey` to handle bookings for Flights, Hotels, and Packages within a single database table.
- **Checkout Flow:** Review details -> Confirm -> Email Notification.
- **Status Tracking:** Bookings track states: `PENDING`, `CONFIRMED`, `CANCELLED`.

- **PDF Generation:** Instant generation of professional PDF tickets containing passenger details, itinerary, and booking reference.

C. Accommodations & Packages (`hotels` & `packages` apps)

- **Hotel Search:** Filter hotels by city and price range. View room types (Single, Double, Suite) and amenities.
- **Holiday Packages:** Curated travel plans with detailed day-by-day itineraries (`Itinerary` model related to `Package`).

Page 4: Database Schema & Data Models

The database is designed with relational integrity in mind. Below are the key data models.

4.1 Users (`accounts`)

- `CustomUser`: Extends `AbstractUser`.
- `email` (EmailField): Primary identifier (unique).
- `phone_number` (CharField): Contact for booking updates.

4.2 Flights (`flights`)

- `Airport`:
- `code` (e.g., "DAC", "JFK").
- `city`, `country`, `name`.
- `Airline`:
- `name`, `logo`.
- `Flight`:
- `airline` (FK to Airline).
- `flight_number`.
- `origin` / `destination` (FKs to Airport).
- `departure_time` / `arrival_time`.
- `price`.

4.3 Bookings (`bookings`)

- `Booking`:
- `user` (FK to CustomUser).
- `content_type` / `object_id`: **Polymorphic relation** linking to either Flight, Room, or Package.
- `status`: Enum (PENDING, CONFIRMED, CANCELLED).
- `booking_date`, `total_price`.

4.4 Inventory (`hotels` & `packages`)

- `Hotel`:
- `name`, `city`, `star_rating`, `image`.
- `Room`:
- `hotel` (FK), `room_type` (Single/Double/Suite), `price_per_night`.
- `Package`:

- `title`, `destination`, `duration_days/nights`, `price`.
- `Itinerary`:
- `package` (FK), `day_number`, `activity`.

Page 5: Key Integrations & Business Logic

5.1 Dynamic Price Categorization

One of the unique logic features in FlyNova is the dynamic price filter in `flights/views.py`.

- **Logic:** The system fetches all flight prices for a search query.
- **Calculation:** It calculates the 33rd and 66th percentiles to dynamically define "Low", "Mid", and "High" tiers for *that specific search result*.
- **Benefit:** This ensures that "High Price" is relative to the current context (e.g., a high price for a domestic flight is different from an international one).

5.2 PDF Ticket Generation

Located in `bookings/views.py`, the `download_ticket_pdf` function uses `reportlab`.

- **Process:**

  1. Creates an in-memory byte buffer.

  2. Draws a canvas using `SimpleDocTemplate`.

  3. Fetches the polymorphic content object (Flight/Hotel/Package) to customize the ticket layout.

  4. Writes User details, Booking ID, and Itinerary into a formatted Table.

  5. Returns the buffer as a `application/pdf` response with `Content-Disposition: attachment`.

5.3 Google OAuth Integration

FlyNova uses `django-allauth` for secure third-party authentication.

- **Flow:** Users click "Sign in with Google" -> Redirect to Google Auth -> Callback to FlyNova -> Account Creation/Login.
- **Configuration:** Managed via `SOCIALACCOUNT_PROVIDERS` in settings, requiring a Client ID and Secret (from Google Cloud Console).
- **Mapping:** Automatically extracts email and name to populate the `CustomUser` model.

Page 6: Installation, Configuration & Deployment

6.1 Prerequisites

- Python 3.8 or higher installed.
- Git for version control.
- Virtual Environment (recommended).

6.2 Local Development Setup

1. Clone the Repository:

`bash

git clone https://github.com/SharefulIslamRony790/flynova.git

cd flynova

`

2. Initialize Environment:

`bash

python -m venv venv

Windows

venv\Scripts\activate

Mac/Linux

source venv/bin/activate

`

3. Install Dependencies:

`bash

pip install -r requirements.txt

`

4. Database Migration:

`bash

python manage.py migrate

`

5. Create Admin User:

`bash

python manage.py createsuperuser

`

6. Run Server:

`bash

python manage.py runserver

Access at http://127.0.0.1:8000

`

6.3 Configuration (.env)

Create a `.env` file (or set environment variables) for sensitive data:

- `SECRET_KEY`: Your unique Django secret.
- `DEBUG`: `True` (Dev) / `False` (Prod).
- `EMAIL_HOST_USER` / `EMAIL_HOST_PASSWORD`: SMTP credentials for booking confirmations.
- `GOOGLE_OAUTH_CLIENT_ID` / `SECRET`: For social login.

6.4 Production Deployment Guidelines

- **WSGI Server:** Use Gunicorn or Waitress instead of `runserver`.
- **Static Files:** Run `python manage.py collectstatic` and serve via Nginx/Apache or WhiteNoise.
- **Database:** Migrate to PostgreSQL for better concurrency handling.
- **Security:** Ensure `DEBUG=False` and set `ALLOWED_HOSTS` to your domain.

End of Project Documentation