Internship Project Documentation

# Bluestock Fintech — IPO Web Application & REST API Development

Intern Name: Neelotpal Sahoo

Supervisor: Yash Kale

Internship Period: 01/06/2025 – 10/07/2025

# Table of Contents

# 1. Introduction

During my internship at Bluestock Fintech, I had the opportunity to contribute to the development of a robust, full-stack web application focused on managing and showcasing Initial Public Offering (IPO) data. This project formed a core part of my internship responsibilities and allowed me to gain hands-on experience in both frontend and backend development using modern web technologies and frameworks.

My primary objective was to design and implement a scalable and user-friendly web interface, along with a fully functional RESTful API to support dynamic data interactions across different components of the application. To achieve this, I actively collaborated with various departments within the organization. I worked alongside the UX/UI design team to convert static wireframes into responsive, interactive user interfaces that adhered to accessibility and usability standards. This included incorporating user feedback into design iterations and ensuring a smooth navigation experience.

On the backend, I engaged closely with backend engineers and database architects to design the application's data models and structure a well-organized database schema. This collaborative effort ensured that the data handling layer was optimized for performance, security, and scalability. I was responsible for developing and integrating various backend functionalities such as user authentication, IPO data management, API endpoints for CRUD operations, and data serialization.

A major focus of the project was ensuring seamless communication between the frontend and backend systems. I implemented secure API calls and handled data fetching and state management efficiently. The application was built to support real-time updates, user-specific views, and administrative access for managing IPO listings.

# 2. Objectives

Frontend Web Portal: Build a responsive interface using Bootstrap 5 and vanilla JavaScript to allow users to:

1. Browse IPOs categorized as Upcoming, Ongoing, and Listed.

2. Search and filter by company name, status, and date range.

3. View detailed IPO pages with real-time computed metrics.

Admin Dashboard: Create secure CRUD functionality enabling administrators to:

1. Add, update, and delete IPO records.

2. Upload and manage related PDF documents (RHP, DRHP) and company logos.

REST API: Expose endpoints with full queryset controls (pagination, filtering, search, ordering).

POSTGRE SQL: To design a reliable backend system using Django and PostgreSQL, providing a scalable and secure foundation for storing and managing IPO-related data.

# 3. Technology Stack

## Backend

SDK: Python 3.12.3

Framework: Django 5.0.6

(Install with: pip install Django)

API: Django REST Framework 3.15.1

(Install with: pip install djangorestframework)

Tools: Postman

## Frontend

Technologies: HTML, CSS, JavaScript

Framework: Bootstrap 5 (CDN)

IDE: Visual Studio Code

## Database

Database: PostgreSQL

Storage: Media for PDFs & logos

# 4. System Architecture

## Step 1: Client Request (Web or API)

The interaction begins when a client (user or external system) sends a request to the server. This can happen in two major ways:

**Web Requests (Browser-based)**

Triggered when a user accesses a webpage or submits a form.

Uses HTTP methods like GET, POST, PUT, and DELETE.

**API Requests (Programmatic)**

Typically triggered by front-end JavaScript using AJAX/fetch or by third-party services integrating via REST API.

Expected content type: application/json.

**Each request contains:**

URL (endpoint)

HTTP method

Headers (e.g., Auth tokens, content type)

Body data (for POST or PUT requests)

## Step 2: Django View / Serializer Processing

Once the request reaches the Django application, the processing logic is handled in multiple stages:

**URL Dispatcher (Routing):**

Django uses urls.py to match the incoming request URL to a specific view function or class-based view.

**View Handling:**

Views act as controllers and contain the logic to process the request.

They call database queries, handle form submissions, or trigger API serializers.

**Serializer (for APIs):**

If the endpoint is API-based, Django REST Framework Serializers are invoked.

**Serializers:**

Validate input data.

Transform complex data types (e.g., querysets) into JSON.

Sanitize and prepare responses.

**Middleware (Pre/Post-processing):**

Middleware components may preprocess requests (e.g., authentication, logging) or post-process responses (e.g., setting headers).

**Key responsibilities:**

Input validation

Business logic execution

Formatting response (HTML or JSON)

## Step 3: Database Query (PostgreSQL)

After view logic processes inputs, it often interacts with the PostgreSQL database via Django's Object-Relational Mapper (ORM).

**PostgreSQL Engine:**

Executes SQL queries under the hood.

Handles ACID-compliant transactions for reliability.

Enforces constraints, relationships, indexing, and schema integrity.

**Data Stored Includes:**

User data

Financial transactions

Uploaded media references

Access logs and configurations

The response (queryset or single object) is passed back to the view for rendering or serialization.

## Step 4: Response – HTML Rendering or JSON Output

Based on the nature of the request, Django prepares one of the following responses:

**HTML Rendering (Template-Based Views):**

For web clients, Django renders a .html file using its template engine.

Dynamic data is injected via context variables.

Uses render(request, "template.html", context).

**JSON Response (API Views):**

For API clients, the response is serialized into JSON using Django REST Framework.

Uses Response(serializer.data) to return data payloads.

**Returned Response:**

Delivered back to the client through HTTP.

May include headers like Content-Type: application/json or text/html.

Security Layer – System-Wide Protections

**1. CSRF Protection (Cross-Site Request Forgery)**

Enabled by Default in Django for all POST requests in web views.

Ensures that state-changing operations originate from trusted sources.

Implemented using the {% csrf_token %} tag in templates and CsrfViewMiddleware.

**2. Token Authentication**

For API requests, Token-based authentication (e.g., JWT or DRF TokenAuth) is used.

Validates users by attaching tokens in headers:

Ensures only authenticated users access protected endpoints.

**3. Input Sanitization and Validation**

All user inputs (both API and form-based) are validated via:

Django Forms (Web)

DRF Serializers (API)

Prevents injection attacks (SQL, XSS) and enforces type and range constraints.

Validation errors are returned with descriptive messages for API clients or form feedback for web users.

**4. Permissions and Access Control**

DRF's permission classes (e.g., IsAuthenticated, IsAdminUser) manage user access.

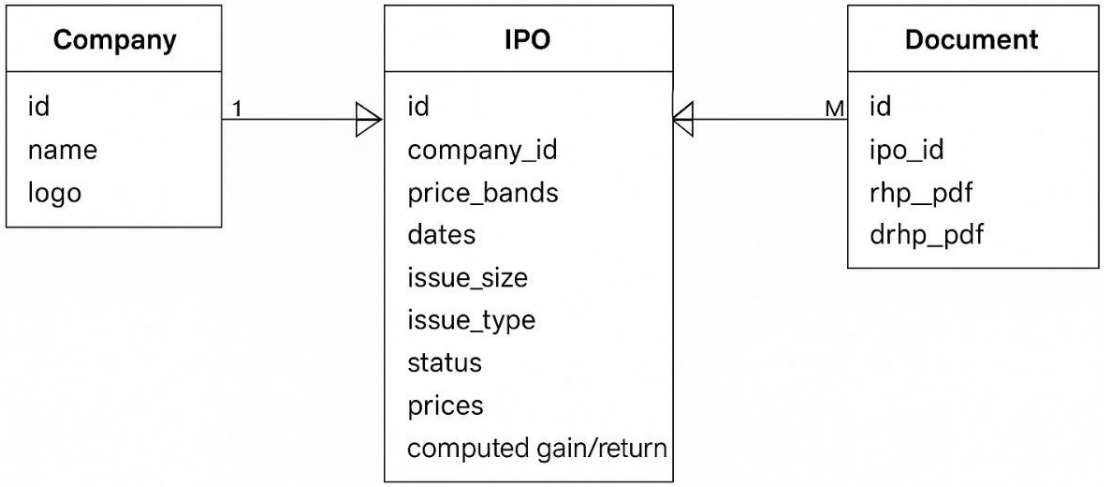
Views and APIs check user roles before executing actions.

**5. HTTPS and Secure Cookies**

In production, SSL/TLS encryption is enforced.

Cookies are marked as Secure, HttpOnly, and SameSite.

# 5. ER DIAGRAM


Company
id
name
logo
1
IPO
id
company_id
price_bands
dates
issue_size
issue_type
status
prices
computed gain/return
M
Document
id
ipo_id
rhp__pdf
drhp_pdf

# 6. Implementation Details

## 6.0 Project Folder Structure:

Bluestock Internship/

| ├── .venv/

| ├── Backend/

│ ├── admin_panel/(__pycache__, migrations, static, templates, __init__.py, admin.py, api_views.py,apps.py,form.py,forms.py,models.py,serializers.py,urls.py,views.py)

├── templates/ admin_panel/(admin_ipos_list.html, base_auth.html, dashboard.html, edit_ipo.html, edit_ipos.html, forgot_password.html, home.html, login.html, manage_ipo_base.html,manage_ipo_dashboard.html,manage_ipos.html,other_admin_pages.html, register_ipo.html, signup.html, upcoming_ipos.html, users.html, view_ipo.html

│ ├── bluestock_project/(__pycache__, templates, asgi.py, settings.py, urls.py, wsgi.py)

│ ├── ipo_app/(admin.py, forms.py, models.py, serializers.py, urls.py, views.py)

├── templates/ ipo_app/(base.html, community.html, contact_us.html, custom_reset_password.html, details.html, footer.html, forgot_password.html, home.html, ipo_api_view.html, ipo_card.html, ipo_list.html, ipos.html, login.html, navbar.html, password_reset_confirm.html, password_reset_email.html, password_reset_subject.txt, signup.html, user_ipo_list.html)

│ ├── manage.py

│ ├── postgre sql file

├── media/

# 6.1 Model Design and Migration

The model design in the Bluestock Fintech platform leverages Django's ORM to represent real-world entities such as Companies, IPOs, and Documents. Each model was crafted with proper field constraints, indexing for performance optimization, and logic encapsulation through computed fields.

## Model Definitions

1. Company Model

Fields: id (UUID), name, logo

**Constraints:**

id: Primary key using UUIDField for global uniqueness.

name: Unique to avoid duplicate companies.

Media handling: logo stored using Django's FileField.

**2. IPO Model**

Fields include:

price_bands, issue_size, issue_type, status, prices, etc.

**Constraints:**

company: Foreign key to Company, with on_delete=CASCADE.

issue_size: Positive number constraint.

**Computed Fields:**

gain and return_percent are calculated automatically in the model's save() method, based on listing price and issue price from prices.

**Indexes:**

Added on status, company, and issue_type to optimize querying and filtering.

**3. Document Model**

Linked via ForeignKey to IPO.

Fields: rhp_pdf, drhp_pdf as file paths for media storage.

## Migration Workflow

To ensure schema synchronization with the database:

**Model Creation and Update:**

Django models are defined/updated in models.py.

**Migration Files Generation:**

python manage.py makemigrations

**Applying Migrations to Database:**

python manage.py migrate

**Schema Validation:**

Verified with python manage.py showmigrations and database inspection tools (like pgAdmin or DBeaver).

## 6.2 REST API Development

The REST API layer was developed using the Django REST Framework (DRF) to enable seamless communication between the frontend, backend, and external systems. The APIs are structured to follow RESTful principles, supporting standard CRUD operations while enforcing strict validation, permission handling, and efficient data querying.

**Serializer Design**

ModelSerializers were created for all core models: Company, IPO, and Document.

**Nested Relationships:**

The IPO serializer nests the Company and Document serializers to return comprehensive, linked data in a single API response.

Example: IPO → includes company name/logo and associated document URLs.

**Custom Validators**

Date validation was implemented to ensure:

Issue open date precedes close date.

Dates are not in the past (for upcoming IPOs).

Validators were added using DRF's validate_<field>() methods or validate() for cross-field logic.

**ViewSet & Routing**

APIs were structured using ModelViewSet, enabling automatic generation of:

list, retrieve, create, update, destroy endpoints.

**Advanced Querying Features:**

Search: Enabled on Company.name, IPO.status, etc.

Filter: Based on fields like issue_type, company_id, or status.

Ordering: By issue_size, computed_gain, or return_percent.

**Endpoint Security**

All API endpoints are secured using DRF's permission system.

IsAdminUser permission was enforced to restrict access to authorized administrative users only.

**Testing & Debugging Tools**

Used Postman to test endpoints for all CRUD operations and edge cases.

**Validated:**

Nested serialization structure

Error messages for invalid inputs

Filtering and search behavior

## 6.3 Frontend Integration

The frontend of the Bluestock Fintech platform was developed using Django's templating engine, enhanced with Bootstrap 5 for responsive design and AJAX for interactive user experiences. The goal was to build an intuitive, fast, and clean UI for browsing and analyzing IPO data without requiring full page reloads.

**Template Structure and Usage**

Django templates were organized within the templates/ directory and designed using modular principles. Key pages include:

**home.html**

Serves as the landing page and IPO listing interface.

Displays all IPOs in a card or tabular layout.

Includes:

Search input

Filters (status, company, issue type)

Summary metrics (e.g., total IPOs, average gain)

**detail.html**

Displays detailed IPO information for a selected record.

**Shows:**

Company name and logo

Price bands, issue dates, issue type

Gain/return metrics

Embedded links for DRHP/RHP documents

**Bootstrap Integration**

Bootstrap 5 (via CDN) was integrated into the base template (base.html) for:

Responsive grid layout (.container, .row, .col)

Cards and tables for IPO display

Navigation bar for page routing

Alerts, badges, and spinners for feedback

**AJAX-Based Search and Filtering**

To improve UX and minimize full-page reloads, JavaScript's Fetch API was used for real-time search and filter functionalities.

**Features:**

Typing in the search box triggers a fetch() request to a backend endpoint with a query parameter.

Filter dropdowns update the IPO list dynamically using GET requests.

The response (JSON) is rendered using JavaScript DOM manipulation.

## Testing and Cross-Browser Compatibility

All pages were tested on Chrome, Firefox, and Edge.

AJAX functionality was verified for valid and edge-case queries.

Bootstrap responsiveness confirmed using device emulation in browser dev tools.

## 6.4 File Management Module

The File Management Module in the Bluestock Fintech system is designed to handle user-uploaded files—such as company logos, RHP (Red Herring Prospectus), and DRHP (Draft Red Herring Prospectus) documents—with security, validation, and proper storage organization.

**Media Configuration**

To manage file uploads, the following settings were configured in settings.py:

MEDIA_URL = '/media/'

MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

MEDIA_ROOT: The absolute path where uploaded files are stored on the server.

MEDIA_URL: The base URL through which media files are accessed publicly.

Example: http://localhost:8000/media/logos/logo.png

Files are served during development via django.views.static.serve and should be handled by a CDN or web server (e.g., NGINX) in production.

**Storage Structure**

Files are organized into dedicated subdirectories for better maintainability and to reflect their associated entity types:

**File Validation**

Custom validation logic was implemented to ensure:

PDFs must be ≤ 5MB

Logos/images must be ≤ 1MB and in accepted formats (PNG, JPG, JPEG)

This validation was handled at the model or form level:

**Security Considerations**

File extensions and MIME types are both checked to prevent spoofed uploads.

Content-Disposition headers are used to control how files are downloaded or opened.

Uploaded files are never executed; they are served as static content.

**Testing**

Verified through Django Admin and API endpoints (via Postman).

**Tested with:**

Oversized files (rejected)

Incorrect formats (rejected)

Valid files (successfully uploaded and rendered)

Logos are rendered in the frontend, and document links are downloadable.

## 6.5 Testing Strategy

To ensure the reliability, correctness, and usability of the Bluestock Fintech platform, a multi-layered testing strategy was employed. This approach combined automated testing for backend logic and API responses with manual testing for user interface validation and cross-browser compatibility.

### 1. Unit Testing

Unit tests were written using Django's built-in unittest framework and the Django REST Framework testing tools. These tests focused on the individual components of the application to ensure their behavior was isolated, predictable, and correct.

### Areas Covered:

**Models:**

IPO gain and return computation logic in save().

Field constraints (e.g., uniqueness, nullability).

**Serializers:**

Nested serialization correctness.

Custom validators (e.g., date ranges, file types).

**API Views:**

Status code responses (200, 400, 403, 404).

Permission enforcement using IsAdminUser.

## 2. Integration Testing (API)

To validate end-to-end functionality of the REST API, Postman was used for:

CRUD operations across Company, IPO, and Document endpoints.

Authentication tests using token headers or session cookies.

Search, filter, and ordering features for IPO listings.

File upload tests for RHP/DRHP PDFs and company logos.

**Test Collection Includes:**

Valid data creation and retrieval

Edge cases: missing fields, invalid file types, broken filters

Unauthorized access attempts

API responses were checked for expected status codes, field structures, nested data, and error messages.

## 3. Frontend & Cross-Browser Testing

Manual testing was conducted across major browsers to ensure consistent rendering and usability of the interface.

**Browsers Tested:**

Google Chrome (latest)

Mozilla Firefox

Microsoft Edge

**Test Scenarios:**

Page responsiveness across screen sizes (desktop, tablet, mobile)

AJAX-based search and filter functionality

File previews and download links

Bootstrap styling and layout consistency

All pages were tested using DevTools device emulation and live preview environments.

## 4. Admin Interface Testing

**Django Admin panel** was tested for:

Form validation (e.g., required fields)

File upload interactions

Inline editing of related models (e.g., IPO under Company)

Permission enforcement

# 7. CODE SNAPSHOTS

## ipo_app/models.py

```
from django.db import models

class IPO(models.Model):
    company_name = models.CharField(max_length=200)
    price_band = models.CharField(max_length=50)
    open_date = models.DateField()
    close_date = models.DateField()
    issue_size = models.CharField(max_length=100)
    issue_type = models.CharField(max_length=50)
    listing_date = models.DateField()
    status = models.CharField(max_length=50)
    logo = models.ImageField(upload_to='logos/', blank=True, null=True)

    def __str__(self):
        return self.company_name
```

## admin_panel/models.py

```
from django.db import models

class IPO(models.Model):
    company_name = models.CharField(max_length=255)
    logo = models.ImageField(upload_to='logos/', blank=True, null=True)
    price_band = models.CharField(max_length=100, blank=True, null=True)
    open_date = models.DateField(blank=True, null=True)
    close_date = models.DateField(blank=True, null=True)
    issue_size = models.CharField(max_length=100, blank=True, null=True)
    issue_type = models.CharField(max_length=50, choices=[
        ('book_building', 'Book Building'),
        ('fixed_price', 'Fixed Price')
    ], blank=True, null=True)
    listing_date = models.DateField(blank=True, null=True)
    status = models.CharField(max_length=50, choices=[
        ('open', 'Open'),
        ('closed', 'Closed'),
        ('upcoming', 'Upcoming')
    ], blank=True, null=True)
```

```
    ipo_price = models.DecimalField(max_digits=10, decimal_places=2,
blank=True, null=True)
    listing_price = models.DecimalField(max_digits=10, decimal_places=2,
blank=True, null=True)
    listing_gain = models.CharField(max_length=100, blank=True, null=True)
    current_return = models.CharField(max_length=100, blank=True,
null=True)
    cmp = models.DecimalField(max_digits=10, decimal_places=2, blank=True,
null=True)

    final_listing_date = models.DateField(blank=True, null=True)
    rhp_link = models.URLField(blank=True, null=True)
    drhp_link = models.URLField(blank=True, null=True)

    def __str__(self):
        return self.company_name
```

## ipo_app/serializers.py

```
from rest_framework import serializers
from .models import IPO

class IPOSerializer(serializers.ModelSerializer):
    class Meta:
        model = IPO
        fields = '__all__'  # or list each field explicitly
```

## admin_panel/serializers.py

```
from rest_framework import serializers
from .models import IPO

class IPOSerializer(serializers.ModelSerializer):
    class Meta:
        model = IPO
        fields = '__all__'
```

## admin_panel/views.py

```
from django.shortcuts import render, redirect
```

```
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
from django.contrib import messages
from .models import IPO
from django.http import JsonResponse
from django.shortcuts import render, get_object_or_404, redirect
from .forms import IPOForm
from django.contrib.auth.signals import user_logged_in
from django.dispatch import receiver
import logging
from rest_framework.permissions import IsAuthenticated
from rest_framework.authentication import SessionAuthentication
from rest_framework import generics
from .models import IPO
from .serializers import IPOSerializer
from django.http import HttpResponse
from datetime import datetime
from decimal import Decimal, InvalidOperation
from .form import CustomUserCreationForm  # Import custom form
import requests

# Home page
def home(request):
    ipos = IPO.objects.all()
    return render(request, 'ipo_app/home.html', {'ipos': ipos})

# Signup
def admin_signup_view(request):
    if request.method == 'POST':
        recaptcha_response = request.POST.get('g-recaptcha-response')
        data = {
            'secret': '6Lfp9nsrAAAAAGOdSVd2HdXYLpJwy07dnfFazmmX',  #
Replace with your reCAPTCHA secret key
            'response': recaptcha_response
        }
        r =
requests.post('https://www.google.com/recaptcha/api/siteverify',
data=data)
        result = r.json()
        print("🔒 reCAPTCHA result:", result)  # Debugging

        form = CustomUserCreationForm(request.POST)
        if form.is_valid() and result.get('success'):
            user = form.save(commit=False)
            user.is_staff = True  # ✅ Make user an admin
```

```
            user.email = form.cleaned_data.get('email')
            user.save()

            # Authenticate and login
            authenticated_user = authenticate(
                username=form.cleaned_data['username'],
                password=form.cleaned_data['password1']
            )
            if authenticated_user is not None:
                login(request, authenticated_user)
                messages.success(request, "Signup successful! Welcome to
Bluestock Admin Panel.")
                return redirect('admin_panel:dashboard')
            else:
                messages.error(request, "✅ Signup worked but
authentication failed. Try logging in manually.")
        else:
            print("⚠ Form errors:", form.errors)  # Debugging
            print("⚠ reCAPTCHA success:", result.get('success'))  #
Debugging
            messages.error(request, "Invalid form data or reCAPTCHA
failed. Please try again.")
    else:
        form = CustomUserCreationForm()

    return render(request, 'admin_panel/signup.html', {'form': form})
# Login
def admin_login_view(request):
    if request.method == 'POST':
        form = AuthenticationForm(request, data=request.POST)
        if form.is_valid():
            user = form.get_user()
            login(request, user)
            return redirect('admin_panel:dashboard')
        else:
            messages.error(request, "Invalid credentials")
    else:
        form = AuthenticationForm()

    return render(request, 'admin_panel/login.html', {'form': form})
# Logout
def logout_view(request):
    logout(request)
    return redirect('admin_panel:login')
```

```python
# Forgot password
def admin_forgot_password_view(request):
    if request.method == 'POST':
        email = request.POST.get('email')
        messages.success(request, 'If this email is registered, a reset
link has been sent.')
        return redirect('admin_panel:forgot_password')
    return render(request, 'admin_panel/forgot_password.html')

logger = logging.getLogger(__name__)

@receiver(user_logged_in)
def log_user_login(sender, request, user, **kwargs):
    logger.info(f"User {user.username} logged in at {user.last_login}")

# IPO List API
def ipo_list_api(request):
    ipos = IPO.objects.all().values('id', 'company_name', 'status',
'open_date', 'close_date', 'logo')
    return JsonResponse(list(ipos), safe=False)

# Admin Dashboard
def admin_dashboard(request):
    ipos = IPO.objects.all()

    return render(request, 'admin_panel/dashboard.html')

# Upcoming IPOs
def admin_upcoming_ipos(request):
    ipos = IPO.objects.filter(status='Upcoming')
    return render(request, 'admin_panel/upcoming_ipos.html')

# Register IPO
def admin_register_ipo(request):
    if request.method == 'POST':
        company_name = request.POST.get('company_name')
        price_band = request.POST.get('price_band')
        issue_size = request.POST.get('issue_size')
        open_date = request.POST.get('open_date')
        close_date = request.POST.get('close_date')
        status = request.POST.get('status')

        IPO.objects.create(
            company_name=company_name,
            price_band=price_band,
```

```
            issue_size=issue_size,
            open_date=open_date,
            close_date=close_date,
            status=status
        )
        return redirect('admin_panel:admin_upcoming_ipos')
    return render(request, 'admin_panel/register_ipo.html')

# Users page in admin
def admin_users(request):
    return render(request, 'admin_panel/users.html')

# List IPOs
def admin_ipos_list(request):
    ipos = IPO.objects.all()
    return render(request, 'admin_panel/admin_ipos_list.html', {'ipos':
ipos})

def get_decimal_or_none(value):
    try:
        return Decimal(value.strip()) if value and value.strip() else None
    except (InvalidOperation, AttributeError):
        return None


def admin_register_ipo(request):
    if request.method == 'POST':
        try:
            company_name = request.POST.get('company_name')
            price_band = request.POST.get('price_band')
            open_date =
datetime.strptime(request.POST.get('open_date').strip(), "%Y-%m-
%d").date()
            close_date =
datetime.strptime(request.POST.get('close_date').strip(), "%Y-%m-
%d").date()
            listing_date =
datetime.strptime(request.POST.get('listing_date').strip(), "%Y-%m-
%d").date()

            issue_size = request.POST.get('issue_size')
            issue_type = request.POST.get('issue_type')
            status = request.POST.get('status')

            ipo_price = get_decimal_or_none(request.POST.get('ipo_price'))
```

```
            listing_price =
get_decimal_or_none(request.POST.get('listing_price'))
            listing_gain =
get_decimal_or_none(request.POST.get('listing_gain'))
            current_return =
get_decimal_or_none(request.POST.get('current_return'))
            cmp = get_decimal_or_none(request.POST.get('cmp'))

            final_listing_date_str =
request.POST.get('final_listing_date')
            final_listing_date = (
                datetime.strptime(final_listing_date_str.strip(), "%Y-%m-
%d").date()
                if final_listing_date_str and
final_listing_date_str.strip() else None
            )

            rhp_link = request.POST.get('rhp_link')
            drhp_link = request.POST.get('drhp_link')
            logo = request.FILES.get('logo')

            ipo = IPO(
                company_name=company_name,
                price_band=price_band,
                open_date=open_date,
                close_date=close_date,
                listing_date=listing_date,
                issue_size=issue_size,
                issue_type=issue_type,
                status=status,
                ipo_price=ipo_price,
                listing_price=listing_price,
                listing_gain=listing_gain,
                current_return=current_return,
                cmp=cmp,
                final_listing_date=final_listing_date,
                rhp_link=rhp_link,
                drhp_link=drhp_link,
                logo=logo
            )
            ipo.save()
            return HttpResponse("IPO registered successfully!")

        except Exception as e:
            return HttpResponse(f"Error: {str(e)}")
```

```
    # GET request - render the form template
    return render(request, 'admin_panel/register_ipo.html')

# View: Upcoming IPOs
def admin_upcoming_ipos(request):
    ipos = IPO.objects.filter(status='Upcoming')
    return render(request, 'admin_panel/upcoming_ipos.html', {
        'ipos': ipos
    })

def admin_manage_ipo(request):
    query = request.GET.get('q', '').strip()

    if query:
        ipos = IPO.objects.filter(company_name__icontains=query)
    else:
        ipos = IPO.objects.all()

    return render(request, 'admin_panel/manage_ipos.html', {'ipos': ipos})

def register_ipo(request):
    if request.method == 'POST':
        form = IPOForm(request.POST, request.FILES)
        if form.is_valid():
            form.save()
            return redirect('admin_panel:admin_manage_ipo')  # Redirect to
IPO list
    else:
        form = IPOForm()

    return render(request, 'admin_panel/register_ipo.html', {'form':
form})

def manage_ipo(request):
    # your logic here
    return render(request, 'admin_panel/manage_ipo.html')

def manage_ipo_view(request):
    ipos = IPO.objects.all()
    return render(request, 'admin_panel/manage_ipo.html', {'ipos': ipos})

def manage_ipo_dashboard(request):
    ipos = IPO.objects.all().order_by('-id')  # Fetch all IPOs
    return render(request, 'admin_panel/manage_ipo.html', {'ipos': ipos})
```

```python
# Edit IPO View
def edit_ipo(request, ipo_id):
    ipo = IPO.objects.get(id=ipo_id)
    if request.method == 'POST':
        ipo.company_name = request.POST.get('company_name')
        ipo.price_band = request.POST.get('price_band')
        ipo.open_date = request.POST.get('open_date')
        ipo.close_date = request.POST.get('close_date')
        ipo.issue_size = request.POST.get('issue_size')
        ipo.issue_type = request.POST.get('issue_type')
        ipo.listing_date = request.POST.get('listing_date')
        ipo.status = request.POST.get('status')
        ipo.ipo_price = request.POST.get('ipo_price')
        ipo.listing_price = request.POST.get('listing_price')
        ipo.listing_gain = request.POST.get('listing_gain')
        ipo.current_return = request.POST.get('current_return')
        ipo.cmp = request.POST.get('cmp')
        ipo.final_listing_date = request.POST.get('final_listing_date')
        ipo.rhp_link = request.POST.get('rhp_link')
        ipo.drhp_link = request.POST.get('drhp_link')
        ipo.save()
        return redirect('admin_panel:manage_ipo')

    return render(request, 'admin_panel/edit_ipo.html', {'ipo': ipo})

# Delete IPO View
def delete_ipo(request, ipo_id):
    ipo = IPO.objects.get(id=ipo_id)
    ipo.delete()
    return redirect('admin_panel:manage_ipo')


def view_ipo(request, ipo_id):
    ipo = get_object_or_404(IPO, id=ipo_id)
    return render(request, 'admin_panel/view_ipo.html', {'ipo': ipo})

def admin_upcoming_ipos(request):
    if request.method == 'POST':
        company_name = request.POST.get('company_name')
        price_band = request.POST.get('price_band')
        open_date = request.POST.get('open_date') or None
        close_date = request.POST.get('close_date') or None
        issue_size = request.POST.get('issue_size')
```

```
        issue_type = request.POST.get('issue_type')
        listing_date = request.POST.get('listing_date') or None
        status = request.POST.get('status')
        ipo_price = request.POST.get('ipo_price') or None
        listing_price = request.POST.get('listing_price') or None
        listing_gain = request.POST.get('listing_gain')
        current_return = request.POST.get('current_return')
        cmp = request.POST.get('cmp')
        rhp_pdf_link = request.POST.get('rhp_pdf_link')
        drhp_pdf_link = request.POST.get('drhp_pdf_link')
        logo = request.FILES.get('logo')

        ipo = IPO(
            company_name=company_name,
            price_band=price_band,
            open_date=open_date,
            close_date=close_date,
            issue_size=issue_size,
            issue_type=issue_type,
            listing_date=listing_date,
            status=status,
            ipo_price=ipo_price,
            listing_price=listing_price,
            listing_gain=listing_gain,
            current_return=current_return,
            cmp=cmp,
            rhp_pdf_link=rhp_pdf_link,
            drhp_pdf_link=drhp_pdf_link,
            logo=logo
        )
        ipo.save()
        messages.success(request, "IPO Registered Successfully!")
        return redirect('admin_upcoming_ipos')

    return render(request, 'admin_panel/upcoming_ipos.html')
```

## ipo_app/views.py

```
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
from django.contrib import messages
from django.http import JsonResponse
import requests
```

```
from rest_framework.permissions import IsAuthenticated
from rest_framework.authentication import SessionAuthentication
from rest_framework import generics
from .serializers import IPOSerializer
from django.shortcuts import render
from rest_framework.decorators import api_view
from rest_framework.response import Response
from rest_framework import viewsets
from rest_framework.permissions import AllowAny
from rest_framework.renderers import JSONRenderer
from .models import IPO
from rest_framework.generics import ListAPIView
from rest_framework.authentication import TokenAuthentication
from rest_framework.decorators import api_view, permission_classes
from rest_framework.views import APIView
from rest_framework import status
from django.contrib.auth.decorators import login_required
from .forms import CustomUserCreationForm  # Import custom form
import requests
from django.contrib.auth.forms import PasswordResetForm
from django.contrib.auth.models import User
from django.template.loader import render_to_string
from django.utils.http import urlsafe_base64_encode
from django.utils.encoding import force_bytes
from django.contrib.auth.tokens import default_token_generator
from django.core.mail import EmailMessage
from django.conf import settings
from django.contrib.auth.hashers import make_password

# Home Page
@login_required(login_url='ipo_app:login')  # Redirect to login if not
logged in
def home(request):
    ipos = IPO.objects.all()
    serializer = IPOSerializer(ipos, many=True)

    upcoming_ipos = [ipo for ipo in serializer.data if
ipo['status'].lower() == 'upcoming']
    ongoing_ipos = [ipo for ipo in serializer.data if
ipo['status'].lower() == 'open']
    listed_ipos = [ipo for ipo in serializer.data if ipo['status'].lower()
== 'closed']

    return render(request, 'ipo_app/home.html', {
        'upcoming_ipos': upcoming_ipos,
```

```
        'ongoing_ipos': ongoing_ipos,
        'listed_ipos': listed_ipos,
    })



from django.contrib.auth import login, authenticate

def signup_view(request):
    if request.method == 'POST':
        recaptcha_response = request.POST.get('g-recaptcha-response')
        data = {
            'secret': '6Lfp9nsrAAAAAGOdSVd2HdXYLpJwy07dnfFazmmX',  #
Replace with your actual reCAPTCHA secret key
            'response': recaptcha_response
        }
        r =
requests.post('https://www.google.com/recaptcha/api/siteverify',
data=data)
        result = r.json()
        print("reCAPTCHA result:", result)  # 🔥 Debugging

        form = CustomUserCreationForm(request.POST)
        if form.is_valid() and result.get('success'):
            user = form.save(commit=False)
            user.email = form.cleaned_data.get('email')
            user.save()

            # Authenticate the user before login
            authenticated_user =
authenticate(username=form.cleaned_data['username'],
password=form.cleaned_data['password1'])
            if authenticated_user is not None:
                login(request, authenticated_user)  # Now Django knows the
backend
                messages.success(request, "Signup successful!")
                return redirect('ipo_app:home')
            else:
                messages.error(request, "Authentication failed. Please try
logging in.")

        else:
            print("Form errors:", form.errors)  # 🔥 Debugging
            print("reCAPTCHA success:", result.get('success'))  # 🔥
Debugging
```

```
            messages.error(request, "Invalid form or reCAPTCHA. Please try
again.")
    else:
        form = CustomUserCreationForm()

    return render(request, 'ipo_app/signup.html', {'form': form})

# Login View
def login_view(request):
    if request.user.is_authenticated:
        return redirect('ipo_app:home')  # ✅ Already logged in? Go home.

    if request.method == 'POST':
        form = AuthenticationForm(request, data=request.POST)
        if form.is_valid():
            user = form.get_user()
            login(request, user)
            return redirect('ipo_app:home')  # ✅ Redirect after login
        else:
            messages.error(request, "Invalid credentials. Please try
again.")
    else:
        form = AuthenticationForm()

    return render(request, 'ipo_app/login.html', {'form': form})


# Logout View
def logout_view(request):
    logout(request)
    return redirect('ipo_app:login')


# Forgot Password View
def forgot_password_view(request):
    if request.method == 'POST':
        email = request.POST.get('email')
        user_qs = User.objects.filter(email=email)
        if user_qs.exists():
            # Get user
            user = user_qs.first()

            # Generate password reset link
            subject = "Reset Your Bluestock Password"
            email_template_name = "ipo_app/password_reset_email.html"
```

```
            context = {
                "email": user.email,
                "domain": request.get_host(),
                "site_name": "Bluestock",
                "uid": urlsafe_base64_encode(force_bytes(user.pk)),
                "user": user,
                "token": default_token_generator.make_token(user),
                "protocol": "https" if request.is_secure() else "http",
            }
            email_body = render_to_string(email_template_name, context)

            # Send Email
            email_msg = EmailMessage(subject, email_body, to=[user.email])
            email_msg.send()

        # Show message whether user exists or not (security best practice)
        messages.success(request, 'If this email is registered, a reset
link has been sent.')
        return redirect('ipo_app:forgot_password')

    return render(request, 'ipo_app/forgot_password.html')

def custom_reset_password_view(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password1 = request.POST.get('password1')
        password2 = request.POST.get('password2')

        if password1 != password2:
            messages.error(request, "Passwords do not match.")
            return render(request, 'ipo_app/custom_reset_password.html')

        try:
            user = User.objects.get(username=username)
            user.password = make_password(password1)  # Hash the password
            user.save()
            messages.success(request, "Password successfully reset. You
can now log in.")
            return redirect('ipo_app:login')
        except User.DoesNotExist:
            messages.error(request, "User not found.")
            return render(request, 'ipo_app/custom_reset_password.html')

    return render(request, 'ipo_app/custom_reset_password.html')
```

```python
# Contact Us View
def contact_us_view(request):
    if request.method == 'POST':
        name = request.POST.get('name')
        email = request.POST.get('email')
        message = request.POST.get('message')
        # Save to database or send email (optional)
        messages.success(request, 'Your message has been sent. Thank
you!')
        return redirect('ipo_app:contact_us')

    return render(request, 'ipo_app/contact_us.html')


# Community View
def community(request):
    if request.method == 'POST':
        if not request.user.is_authenticated:
            messages.warning(request, "Please login to post.")
            return redirect('ipo_app:login')

        content = request.POST.get('content')
        # Optional: Save post
        messages.success(request, 'Your post has been submitted.')
        return redirect('ipo_app:community')

    return render(request, 'ipo_app/community.html')

class IpoViewSet(viewsets.ModelViewSet):
    queryset = IPO.objects.all()
    serializer_class = IPOSerializer


# API: IPO List
@api_view(['GET'])
@permission_classes([AllowAny])
def ipo_list_api(request):
    ipos = IPO.objects.all()
    serializer = IPOSerializer(ipos, many=True)
    return Response(serializer.data)
```

```
def ipo_list_view(request):
    return render(request, 'ipo_app/ipo_list.html')

def user_ipo_list(request):
    return render(request, 'ipo_app/user_ipo_list.html')


def ipo_api_view(request):
    return render(request, 'ipo_app/ipo_api_view.html')


class IPOListAPIView(ListAPIView):
    queryset = IPO.objects.all()
    serializer_class = IPOSerializer

    # API View to get list of IPOs (authenticated)
class IPOListAPIView(generics.ListAPIView):
    queryset = IPO.objects.all()
    serializer_class = IPOSerializer
    authentication_classes = [TokenAuthentication]
    permission_classes = [IsAuthenticated]

class IPOCreateView(APIView):
    def post(self, request):
        serializer = IPOSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data,
status=status.HTTP_201_CREATED)
        return Response(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)


        serializer = IPOSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data,
status=status.HTTP_201_CREATED)
        return Response(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)
```

## ipo_app/urls.py

```
# ipo_app/urls.py
```

```python
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from .views import IpoViewSet
from rest_framework.authtoken.views import obtain_auth_token
from . import views
from django.contrib.auth import views as auth_views
from django.urls import reverse_lazy

app_name = 'ipo_app'

# Register the API ViewSet
router = DefaultRouter()
router.register(r'ipos', IpoViewSet, basename='ipo')

urlpatterns = [
    # Website routes
    path('', views.home, name='home'),
    path('login/', views.login_view, name='login'),
    path('signup/', views.signup_view, name='signup'),
    path('logout/', views.logout_view, name='logout'),
    path('contact-us/', views.contact_us_view, name='contact_us'),
    path('community/', views.community, name='community'),
    path('ipos/', views.ipo_list_view, name='ipo_list'),


    # API endpoints
    path('api/', include(router.urls)),  # This handles /api/ipos/
    path('api/token-auth/', obtain_auth_token),
    path('auth/', include('social_django.urls', namespace='social')),
    # Forgot password flow
    path('reset-password/', views.custom_reset_password_view,
name='reset_password'),
    path('forgot-password/', views.custom_reset_password_view,
name='forgot_password'),
    path('reset-password/<uidb64>/<token>/',
auth_views.PasswordResetConfirmView.as_view(
    template_name='ipo_app/password_reset_confirm.html',
    success_url='/login/'
    ), name='password_reset_confirm'),
]
```

## admin_panel/urls.py

```
from django.urls import path
from . import views
from django.urls import path, include
from django.contrib import admin
from django.urls import path, include
from django.contrib.auth import views as auth_views
from .api_views import IPOListAPIView
from .views import admin_register_ipo

app_name = 'admin_panel'

urlpatterns = [
    path('', views.home, name='home'),
    path('signup/', views.admin_signup_view, name='signup'),
    path('login/', views.admin_login_view, name='login'),
    path('forgot-password/', views.admin_forgot_password_view,
name='forgot_password'),
    path('dashboard/', views.admin_dashboard, name='dashboard'),
    path('logout/',  views.logout_view,         name='admin_logout'),

    path('api/ipo/', views.ipo_list_api, name='ipo_list_api'),

    # path('upcoming-ipos/', views.admin_upcoming_ipos,
name='admin_upcoming_ipos'),
    path('upcoming-ipos/', views.admin_upcoming_ipos,
name='admin_upcoming_ipos'),
    path('upcoming-ipos/edit/<int:ipo_id>/', views.edit_ipo,
name='edit_ipo'),
    path('upcoming-ipos/delete/<int:ipo_id>/', views.delete_ipo,
name='delete_ipo'),
    path('register-ipo/', views.admin_register_ipo,
name='admin_register_ipo'),
    path('users/', views.admin_users, name='admin_users'),
    path('home/', views.home, name='home'),
    path('logout/', views.logout_view, name='logout'),
    path('manage-ipo/', views.admin_manage_ipo, name='admin_manage_ipo'),
    path('manage-ipo/ipo-information/', views.register_ipo,
name='register_ipo'),
    path('register-ipo/', views.register_ipo, name='admin_register_ipo'),
    path('manage-ipo/', views.manage_ipo, name='manage_ipo'),
    path('edit-ipo/<int:ipo_id>/', views.edit_ipo, name='edit_ipo'),
    path('delete-ipo/<int:ipo_id>/', views.delete_ipo, name='delete_ipo'),
    path('manage-ipo/', views.manage_ipo_view, name='admin_manage_ipo'),
```

```
    path('view-ipo/<int:ipo_id>/', views.view_ipo, name='view_ipo'),
    path('admin/manage-ipos/', views.admin_manage_ipo,
name='admin_manage_ipo'),
    path('api/ipos/', IPOListAPIView.as_view(), name='ipo_list_api'),  #
<-- Your API endpoint
    path('admin-panel/register-ipo/', admin_register_ipo,
name='register_ipo'),
]
```

## ipo_app/forms.py

```
# forms.py
from django import forms
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User

class CustomUserCreationForm(UserCreationForm):
    email = forms.EmailField(required=True, help_text="Required. Enter a
valid email address.")

    class Meta:
        model = User
        fields = ('username', 'email', 'password1', 'password2')

    def clean_email(self):
        email = self.cleaned_data.get('email')
        if User.objects.filter(email=email).exists():
            raise forms.ValidationError("This email is already
registered.")
        return email
```

## admin_panel/form.py

```
# forms.py
from django import forms
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User

class CustomUserCreationForm(UserCreationForm):
    email = forms.EmailField(required=True,
widget=forms.EmailInput(attrs={
        'class': 'form-control',
```

```
        'placeholder': ''
    }))

    class Meta:
        model = User
        fields = ('username', 'email', 'password1', 'password2')
```

## ipo_app/templates/ipo_app/details.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Bluestock IPO Details</title>

  <!-- Bootstrap 5 -->
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.
css" rel="stylesheet" />

  <!-- Custom CSS -->
  <link rel="stylesheet" href="{% static 'css/style.css' %}" />
</head>
<body>

  <!-- Navbar -->
  <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
    <div class="container">
      <a class="navbar-brand fw-bold" href="/">
        <img src="{% static 'images/logo.png' %}" alt="Bluestock Logo"
height="40" class="me-2" />
        Bluestock
      </a>
    </div>
  </nav>

  <!-- Main content -->
  <div class="container my-5">

    <!-- Back button -->
    <a href="/" class="btn btn-secondary mb-4">← Back to Home</a>
```

```
    <!-- IPO details -->
    <div id="ipoDetails" class="card p-4 shadow">
      <!-- JavaScript will insert IPO details here -->
    </div>

  </div>

  <!-- Footer -->
  <footer class="bg-primary text-white text-center py-3">
    <div class="container">
      © 2025 Bluestock Fintech. All Rights Reserved.
    </div>
  </footer>

  <!-- Script -->
  <script>
    const ipoId =
window.location.pathname.split('/').filter(Boolean).pop();

    async function fetchIPODetails() {
      const res = await fetch(`/api/ipo/${ipoId}/`);
      const ipo = await res.json();

      const html = `
        <div class="text-center mb-4">
          <img src="/media/${ipo.logo}" alt="${ipo.company_name} Logo"
class="media-logo mb-3">
          <h3>${ipo.company_name}</h3>
        </div>

        <ul class="list-group mb-4">
          <li class="list-group-item"><strong>Status:</strong>
${ipo.status}</li>
          <li class="list-group-item"><strong>Price Band:</strong>
${ipo.price_band}</li>
          <li class="list-group-item"><strong>Open Date:</strong>
${ipo.open_date}</li>
          <li class="list-group-item"><strong>Close Date:</strong>
${ipo.close_date}</li>
          <li class="list-group-item"><strong>Issue Size:</strong>
${ipo.issue_size}</li>
          <li class="list-group-item"><strong>Issue Type:</strong>
${ipo.issue_type}</li>
          <li class="list-group-item"><strong>Listing Date:</strong>
${ipo.listing_date || '-'}</li>
```

```
          <li class="list-group-item"><strong>IPO Price:</strong>
₹${ipo.ipo_price || '-'}</li>
          <li class="list-group-item"><strong>Listing Price:</strong>
₹${ipo.listing_price || '-'}</li>
          <li class="list-group-item"><strong>Current Market
Price:</strong> ₹${ipo.current_market_price || '-'}</li>
          <li class="list-group-item"><strong>Listing Gain:</strong>
${ipo.listing_gain ? ipo.listing_gain + '%' : '-'}</li>
          <li class="list-group-item"><strong>Current Return:</strong>
${ipo.current_return ? ipo.current_return + '%' : '-'}</li>
        </ul>

        <div class="d-flex gap-3 flex-wrap">
          ${ipo.rhp_pdf ? `<a href="/media/${ipo.rhp_pdf}" class="btn btn-
outline-primary" target="_blank">📄 Download RHP</a>` : ''}
          ${ipo.drhp_pdf ? `<a href="/media/${ipo.drhp_pdf}" class="btn
btn-outline-secondary" target="_blank">📄 Download DRHP</a>` : ''}
        </div>
      `;

      document.getElementById('ipoDetails').innerHTML = html;
    }

    fetchIPODetails();
  </script>

</body>
</html>
```

## ipo_app/templates/ipo_app/home.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Bluestock IPO Tracker</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.
css" rel="stylesheet">
</head>
<body>
```

```
  <style>
    body {
  height: 100%;
}

body {
  display: flex;
  flex-direction: column;
}

main {
  flex: 1;
}

/* Optionally: Some spacing for content */
main {
  padding-bottom: 2rem;
}

/* General body styles */
body {
  background-color: #f8f9fa;
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  color: #333;
}

/* Headers */
h1, h2, h3 {
  color: #0d6efd;
  font-weight: 600;
}

/* Navbar brand */
.navbar-brand {
  font-size: 1.5rem;
  display: flex;
  align-items: center;
}

.navbar-brand img {
  border-radius: 8px;
  background-color: white;
  padding: 4px;
}
```

```
    /* Global Styles */
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: #f8f9fa;
}

/* Login/Signup Form Container */
.auth-form {
  max-width: 600px;
  margin: 60px auto;
  padding: 90px;
  background: #ffffff;
  border-radius: 10px;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
}

/* Form Title */
.auth-form h2 {
  margin-bottom: 25px;
  color: #0d6efd;
  font-weight: bold;
  text-align: center;
}

/* Input Fields */
.auth-form .form-control {
  border-radius: 8px;
  padding: 12px;
}

/* Submit Button */
.auth-form .btn-primary {
  width: 100%;
  padding: 12px;
  font-size: 1.1rem;
  border-radius: 8px;
  margin-top: 15px;
}

/* Link below form (eg: Already have account?) */
.auth-form .form-text {
  margin-top: 15px;
  text-align: center;
}
```

```
/* Optional: Small fade animation */
.auth-form {
  animation: fadeIn 0.6s ease-in-out;
}

@keyframes fadeIn {
  from { opacity: 0; transform: translateY(20px); }
  to { opacity: 1; transform: translateY(0); }
}
    /* IPO cards */
.card {
  border-radius: 12px;
  transition: transform 0.2s ease;
}

.card:hover {
  transform: translateY(-5px);
}

.card-title {
  font-size: 1.25rem;
  font-weight: bold;
  color: #212529;
}

.card-img-top {
  object-fit: contain;
  height: 180px;
  padding: 15px;
  background-color: white;
  border-bottom: 1px solid #eee;
}

.card-body p {
  margin-bottom: 8px;
  font-size: 0.95rem;
}

/* Search bar */
#searchInput {
  max-width: 400px;
  border-radius: 30px;
  padding: 10px 20px;
  border: 1px solid #ccc;
```

```
}

.tabs {
    margin-bottom: 20px;
}

.tab-button {
    padding: 10px 15px;
    border: none;
    background: none;
    font-size: 16px;
    cursor: pointer;
    color: #007bff;
}

.tab-button.active {
    border-bottom: 2px solid #007bff;
    font-weight: bold;
}


.search-input {
    padding: 8px;
    width: 250px;
    margin-bottom: 10px;
    border: 1px solid #ccc;
    border-radius: 4px;
}


/* Media logo on detail page */
.media-logo {
  width: 150px;
  height: auto;
  border-radius: 10px;
  box-shadow: 0 4px 8px rgba(0,0,0,0.1);
  background: white;
  padding: 10px;
  margin-bottom: 20px;
}

/* Download buttons */
.btn-outline-primary,
.btn-outline-secondary {
  font-weight: 500;
```

```
  font-size: 0.9rem;
  padding: 10px 15px;
  border-radius: 8px;
}
   /* List group */
.list-group-item {
  font-size: 0.95rem;
}

.footer {
  background-color: #f8f8f8;
  font-family: 'Segoe UI', sans-serif;
  padding: 30px 20px;
  color: #333;
  font-size: 14px;
}

.footer-top {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(140px, 1fr));
  gap: 30px;
  margin-bottom: 30px;
}

.footer-column h4 {
  font-weight: 600;
  margin-bottom: 10px;
  color: #111;
}

.footer-column ul {
  list-style: none;
  padding: 0;
}

.footer-column ul li {
  margin-bottom: 6px;
  color: #444;
}

.footer-middle {
  display: flex;
  flex-direction: column;
  gap: 25px;
  border-top: 1px solid #ccc;
```

```
  padding-top: 20px;
}


.footer-content {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  flex-wrap: wrap;
  gap: 30px;
}

.footer-left {
  max-width: 280px;
  flex: 1;
}

.footer-right {
  flex: 2;
  min-width: 320px;
}

.footer-right p {
  margin-bottom: 12px;
  font-size: 14px;
  color: #444;
  line-height: 1.6;
}

.footer-right .disclaimer {
  font-weight: bold;
  color: #000;
}


.social-icons a img {
  width: 24px;
  margin-right: 20px;
  vertical-align: middle;
}

.footer-logo {
  width: 120px;
  margin-bottom: 10px;
}
```

```
.startup-img {
  width: 100px;
  margin-top: 10px;
}

.footer-info p {
  margin: 6px 0;
  color: #444;
}

.footer-text p {
  margin-bottom: 10px;
  color: #555;
}

.footer-text .disclaimer {
  font-weight: bold;
  color: #000;
}

.footer-bottom {
  border-top: 1px solid #ccc;
  margin-top: 20px;
  padding-top: 10px;
  display: flex;
  justify-content: space-between;
  flex-wrap: wrap;
  font-size: 13px;
  color: #777;
}

.faq-section {
  width: 100%;
  background-color: #f9f9f9;
  padding: 40px 0;
  font-family: 'Segoe UI', sans-serif;
}

.faq-section h2 {
  font-size: 28px;
  font-weight: bold;
  margin-bottom: 8px;
  text-align: center;
}
```

```
.faq-section p {
  font-size: 15px;
  color: #555;
  margin-bottom: 24px;
  text-align: center;
}

.faq-container {
  max-width: 1000px;
  margin: 0 auto;
  border-top: 1px solid #ddd;
}

.faq-item {
  width: 100%;
  border-bottom: 1px solid #ddd;
  background-color: #fff;
}

.faq-question {
  display: flex;
  justify-content: space-between;
  align-items: center;
  width: 100%;
  padding: 18px 24px;
  background: #fff;
  border: none;
  cursor: pointer;
  font-size: 18px;
  font-weight: 600;
  text-align: left;
  color: #333;
}

.faq-answer {
  display: none;
  padding: 0 24px 18px;
  font-size: 15px;
  color: #444;
  background: #fff;
  text-align: left;  /* Make sure text is left aligned */
}

.faq-answer ul {
```

```
  margin: 0;
  padding-left: 20px;
}

.faq-item.active .faq-answer {
  display: block;
}

.faq-item .icon {
  font-size: 24px;
  margin-left: 12px;
  color: #007bff;
  font-weight: bold;
  transition: transform 0.3s ease;
}

.faq-item.active .icon {
  transform: rotate(180deg);
}
</style>

  <!-- NAVBAR -->
  <nav class="navbar navbar-expand-lg bg-white shadow-sm py-2">
    <div class="container">
      <a class="navbar-brand fw-bold d-flex align-items-center" href="/">
        <img src="/static/images/logo.png" alt="Logo" height="40"
class="me-2" />
      </a>
      <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#mainNavbar">
        <span class="navbar-toggler-icon"></span>
      </button>

      <div class="collapse navbar-collapse" id="mainNavbar">
        <ul class="navbar-nav mx-auto">
          <li class="nav-item"><a href="#" class="nav-
link">PRODUCTS</a></li>
          <li class="nav-item"><a href="#" class="nav-
link">PRICING</a></li>
          <li class="nav-item"><a href="/community" class="nav-
link">COMMUNITY</a></li>
          <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle" data-bs-toggle="dropdown"
href="#">MEDIA</a>
            <ul class="dropdown-menu">
```

```
              <li><a class="dropdown-item" href="#">Photos</a></li>
              <li><a class="dropdown-item" href="#">Videos</a></li>
            </ul>
          </li>
          <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle" data-bs-toggle="dropdown"
href="#">SUPPORT</a>
            <ul class="dropdown-menu">
              <li><a class="dropdown-item" href="#">Help Center</a></li>
              <li><a class="dropdown-item" href="/contact-us">Contact
Us</a></li>
            </ul>
          </li>
        </ul>
        <!-- Right side: Login / Signup / Grid -->
      <div class="d-flex justify-content-end align-items-center gap-2">
  {% if user.is_authenticated %}
    <span class="me-2 text-dark">Hello, {{ user.username }}</span>
    <a href="{% url 'ipo_app:logout' %}" class="btn btn-sm btn-
danger">Logout</a>
  {% else %}
    <a href="{% url 'ipo_app:login' %}" class="btn btn-sm btn-outline-
primary">Sign In</a>
    <a href="{% url 'ipo_app:signup' %}" class="btn btn-sm btn-
primary">Sign Up Now</a>
  {% endif %}
</div>
      </div>
    </div>
  </nav>

  <!-- Header with Typewriter -->
<header class="py-5 bg-light text-center">
  <div class="container">
    <h1 class="display-5">
      <span id="typed-text"></span>
    </h1>
    <p class="lead">Track Upcoming, Ongoing, and Listed IPOs — Download
RHP & DRHP and view performance.</p>
  </div>
</header>

      <!-- Typed.js -->
<script src="https://cdn.jsdelivr.net/npm/typed.js@2.0.12"></script>
<script>
```

```
  var typed = new Typed('#typed-text', {
  strings: [
    'Welcome to Bluestock IPO Tracker',
    'Find Latest IPO Updates',
    'Track and Download RHP/DRHP'
  ],
  typeSpeed: 50,
  backSpeed: 30,
  backDelay: 2000,  // reduced delay
  startDelay: 500,
  loop: true,
  smartBackspace: true  // smoother typing
});
</script>

  <main class="container my-5">
    <div class="d-flex justify-content-between align-items-center mb-4">
      <h2>All IPOs</h2>
      <input type="text" id="searchInput" class="form-control w-50"
placeholder="Search IPO by company name" />
    </div>

    <!-- IPO Tabs -->
    <ul class="nav nav-tabs mb-3" id="ipoTabs" role="tablist">
      <li class="nav-item" role="presentation">
        <button class="nav-link active" id="upcoming-tab" data-bs-
toggle="tab" data-bs-target="#upcoming" type="button" role="tab">Upcoming
IPO</button>
      </li>
      <li class="nav-item" role="presentation">
        <button class="nav-link" id="ongoing-tab" data-bs-toggle="tab"
data-bs-target="#ongoing" type="button" role="tab">Ongoing IPO</button>
      </li>
      <li class="nav-item" role="presentation">
        <button class="nav-link" id="listed-tab" data-bs-toggle="tab"
data-bs-target="#listed" type="button" role="tab">Listed IPO</button>
      </li>
    </ul>

    <!-- IPO Tab Content -->
    <div class="tab-content">
      <div class="tab-pane fade show active" id="upcoming" role="tabpanel"
aria-labelledby="upcoming-tab">
        <div class="row" id="upcomingCards"></div>
      </div>
```

```
      <div class="tab-pane fade" id="ongoing" role="tabpanel" aria-
labelledby="ongoing-tab">
        <div class="row" id="ongoingCards"></div>
      </div>
      <div class="tab-pane fade" id="listed" role="tabpanel" aria-
labelledby="listed-tab">
        <div class="row" id="listedCards"></div>
      </div>
    </div>
  </main>

  <!-- Bootstrap JS -->
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle
.min.js"></script>

  <!-- IPO Script -->
  <script>
    async function fetchAllIPOs() {
      let ipos = [];
      let url = "/api/ipos/";
      const headers = {
        'Authorization': 'Token 95e5fce568a5ad76243dd7aea61a6050ce0eb06c',
        'Content-Type': 'application/json'
      };

      try {
        while (url) {
          const resp = await fetch(url, { headers });
          const res = await resp.json();

          if (Array.isArray(res.results)) {
            ipos.push(...res.results);
            url = res.next;
          } else if (Array.isArray(res)) {
            ipos = res;
            url = null;
          } else {
            console.error("Unexpected API response format:", res);
            break;
          }
        }

        console.log("✅ All IPOs fetched:", ipos);
        renderIPOs(ipos);
```

```
      } catch (e) {
        console.error("❌ Error loading IPOs:", e);
      }
    }

    function determineStatus(ipo) {
      const today = new Date();
      const open = new Date(ipo.open_date);
      const close = new Date(ipo.close_date);

      if (isNaN(open) || isNaN(close)) return "upcoming";
      if (today < open) return "upcoming";
      if (today >= open && today <= close) return "ongoing";
      return "listed";
    }

    function renderIPOs(ipos) {
      const containers = {
        upcoming: document.getElementById('upcomingCards'),
        ongoing: document.getElementById('ongoingCards'),
        listed: document.getElementById('listedCards')
      };

      Object.values(containers).forEach(el => el.innerHTML = '');

      ipos.forEach(ipo => {
        const status = determineStatus(ipo);
        const logo = ipo.logo || '/media/logos/default-logo.png';

        const card = `
          <div class="col-md-4 mb-4 ipo-card" data-
name="${ipo.company_name.toLowerCase()}">
            <div class="card h-100 shadow-sm">
              <img src="${logo}" class="card-img-top"
alt="${ipo.company_name}" onerror="this.src='/media/logos/default-
logo.png'">
              <div class="card-body">
                <h5 class="card-title text-
primary">${ipo.company_name}</h5>
                <p><strong>Price Band:</strong> ${ipo.price_band}</p>
                <p><strong>Open:</strong> ${ipo.open_date} |
<strong>Close:</strong> ${ipo.close_date}</p>
                <p><strong>Issue Size:</strong> ${ipo.issue_size ||
'N/A'}</p>
```

```
                <p><strong>Issue Type:</strong> ${ipo.issue_type ||
'N/A'}</p>
                <p><strong>Listing Date:</strong> ${ipo.listing_date ||
'N/A'}</p>
                <p><strong>RHP:</strong>
                  <a href="${ipo.rhp_pdf}" target="_blank" class="btn btn-
sm btn-outline-primary ms-2">Download</a>
                </p>
                <p><strong>DRHP:</strong>
                  <a href="${ipo.drhp_pdf}" target="_blank" class="btn
btn-sm btn-outline-primary ms-2">Download</a>
                </p>
                <span class="badge bg-${status === 'upcoming' ? 'warning'
: status === 'ongoing' ? 'success' : 'secondary'} text-
dark">${status}</span>
              </div>
            </div>
          </div>`;

        containers[status].insertAdjacentHTML('beforeend', card);
      });

      Object.entries(containers).forEach(([key, el]) => {
        if (!el.innerHTML.trim()) {
          el.innerHTML = `<p class="text-muted">No ${key} IPOs
available.</p>`;
        }
      });

      // 🔧 Ensure visible tab-pane is correctly shown
      setTimeout(() => {
        const activeBtn = document.querySelector(".nav-link.active");
        if (activeBtn) {
          const targetId = activeBtn.getAttribute("data-bs-target");
          const targetPane = document.querySelector(targetId);
          if (targetPane) {
            document.querySelectorAll(".tab-pane").forEach(p =>
p.classList.remove("show", "active"));
            targetPane.classList.add("show", "active");
          }
        }
      }, 100);
    }

    document.addEventListener("DOMContentLoaded", () => {
```

```
      fetchAllIPOs();

      // Search
      const searchInput = document.getElementById('searchInput');
      if (searchInput) {
        searchInput.addEventListener('input', function () {
          const query = this.value.trim().toLowerCase();
          document.querySelectorAll('.ipo-card').forEach(card => {
            const name = card.getAttribute('data-name');
            card.style.display = name.includes(query) ? '' : 'none';
          });
        });
      }
    });
  </script>

<!-- Optional: Search Filter -->
<script>
document.getElementById('searchInput').addEventListener('input', function
() {
  const query = this.value.toLowerCase();
  const cards = document.querySelectorAll('.ipo-card');
  cards.forEach(card => {
    const name = card.getAttribute('data-name');
    card.style.display = name.includes(query) ? '' : 'none';
  });
});
</script>



<!-- Footer -->
{% include 'ipo_app/footer.html' %}


<!-- FAQ Section -->
<section class="faq-section my-5 py-4 px-3 bg-light rounded">
  <h2 class="mb-3 fw-bold text-center">Frequently Asked Questions?</h2>
  <p class="text-center text-muted mb-4">
    Find answers to common questions that come in your mind related to
IPO.
  </p>

  <div class="faq-container">
```

```
    <div class="faq-item active">
      <button class="faq-question w-100 text-start d-flex justify-content-
between align-items-center p-3 border-0 bg-white rounded shadow-sm">
        How to Subscribe to an IPO?
        <span class="icon fs-4">-</span>
      </button>
      <div class="faq-answer p-3">
        <ul>
          <li>Step 1: Login to your respective service provider.</li>
          <li>Step 2: Click on the IPO button.</li>
          <li>Step 3: Select the IPO you want to bid and enter the
relevant details.</li>
          <li>Step 4: Your subscription will be completed once you make
the payment or give permission.</li>
        </ul>
      </div>
    </div>

    <div class="faq-item">
      <button class="faq-question w-100 text-start d-flex justify-content-
between align-items-center p-3 border-0 bg-white rounded shadow-sm">
        Should I buy an IPO first day?
        <span class="icon fs-4">+</span>
      </button>
      <div class="faq-answer p-3"></div>
    </div>

    <div class="faq-item">
      <button class="faq-question w-100 text-start d-flex justify-content-
between align-items-center p-3 border-0 bg-white rounded shadow-sm">
        How do you know if an IPO is good?
        <span class="icon fs-4">+</span>
      </button>
      <div class="faq-answer p-3"></div>
    </div>

    <div class="faq-item">
      <button class="faq-question w-100 text-start d-flex justify-content-
between align-items-center p-3 border-0 bg-white rounded shadow-sm">
        How to check IPO start date?
        <span class="icon fs-4">+</span>
      </button>
      <div class="faq-answer p-3"></div>
    </div>
```

```
    <div class="faq-item">
      <button class="faq-question w-100 text-start d-flex justify-content-
between align-items-center p-3 border-0 bg-white rounded shadow-sm">
        What is issue size?
        <span class="icon fs-4">+</span>
      </button>
      <div class="faq-answer p-3"></div>
    </div>

    <div class="faq-item">
      <button class="faq-question w-100 text-start d-flex justify-content-
between align-items-center p-3 border-0 bg-white rounded shadow-sm">
        How many shares in a lot?
        <span class="icon fs-4">+</span>
      </button>
      <div class="faq-answer p-3"></div>
    </div>

    <div class="faq-item">
      <button class="faq-question w-100 text-start d-flex justify-content-
between align-items-center p-3 border-0 bg-white rounded shadow-sm">
        How is the lot size calculated?
        <span class="icon fs-4">+</span>
      </button>
      <div class="faq-answer p-3"></div>
    </div>

    <div class="faq-item">
      <button class="faq-question w-100 text-start d-flex justify-content-
between align-items-center p-3 border-0 bg-white rounded shadow-sm">
        Who decides the IPO price band?
        <span class="icon fs-4">+</span>
      </button>
      <div class="faq-answer p-3"></div>
    </div>

    <div class="faq-item">
      <button class="faq-question w-100 text-start d-flex justify-content-
between align-items-center p-3 border-0 bg-white rounded shadow-sm">
        What is IPO GMP?
        <span class="icon fs-4">+</span>
      </button>
      <div class="faq-answer p-3"></div>
    </div>
```

```
    <div class="faq-item">
      <button class="faq-question w-100 text-start d-flex justify-content-
between align-items-center p-3 border-0 bg-white rounded shadow-sm">
        How many lots should I apply for IPO?
        <span class="icon fs-4">+</span>
      </button>
      <div class="faq-answer p-3"></div>
    </div>

  </div>
</section>

<script>
  document.addEventListener("DOMContentLoaded", function () {
    const faqItems = document.querySelectorAll(".faq-item");

    faqItems.forEach((item) => {
      const button = item.querySelector(".faq-question");
      const answer = item.querySelector(".faq-answer");
      const icon = item.querySelector(".icon");

      button.addEventListener("click", () => {
        const isActive = item.classList.contains("active");

        // Close all items first
        faqItems.forEach((el) => {
          el.classList.remove("active");
          el.querySelector(".faq-answer").style.display = "none";
          el.querySelector(".icon").textContent = "+";
        });

        // If not already active, activate it
        if (!isActive) {
          item.classList.add("active");
          answer.style.display = "block";
          icon.textContent = "-";
        }
      });

      // Initialize state
      if (!item.classList.contains("active")) {
        answer.style.display = "none";
        icon.textContent = "+";
      } else {
        answer.style.display = "block";
```

```
        icon.textContent = "-";
      }
    });
  });
</script>




<!-- Footer -->
<footer class="footer">
  <div class="footer-top">
    <div class="footer-column">
      <h4>Resources</h4>
      <ul>
        <li>Trading View</li>
        <li>NSE Holidays</li>
        <li>e-Voting CDSL</li>
        <li>e-Voting NSDL</li>
        <li>Market Timings</li>
      </ul>
    </div>
    <div class="footer-column">
      <h4>Company</h4>
      <ul>
        <li>Careers</li>
        <li>Contact Us</li>
        <li>About Us</li>
        <li>Community</li>
        <li>Blogs</li>
      </ul>
    </div>
    <div class="footer-column">
      <h4>Offerings</h4>
      <ul>
        <li>Compare Broker</li>
        <li>Fin Calculators</li>
        <li>IPO</li>
        <li>All Brokers</li>
        <li>Products</li>
      </ul>
    </div>
    <div class="footer-column">
      <h4>Links</h4>
```

```
      <ul>
        <li>Shark Investor</li>
        <li>Mutual Funds</li>
        <li>Sitemap</li>
        <li>Indian indices</li>
        <li>Bug Bounty Program</li>
      </ul>
    </div>
    <div class="footer-column">
      <h4>Policy</h4>
      <ul>
        <li>Terms & Conditions</li>
        <li>Privacy Policy</li>
        <li>Refund Policy</li>
        <li>Disclaimer</li>
        <li>Trust & Security</li>
      </ul>
    </div>
  </div>

  <div class="footer-middle">
  <div class="footer-content">
    <!-- Left section: social + logo + info -->
    <div class="footer-left">
      <div class="social-icons">
        <a href="#"><img src="{% static 'images/logo_2.png' %}"
alt="Facebook"></a>
        <a href="#"><img src="{% static 'images/logo_3.png' %}"
alt="LinkedIn"></a>
        <a href="#"><img src="{% static 'images/logo_5.png' %}"
alt="Twitter"></a>
        <a href="#"><img src="{% static 'images/logo_4.png' %}"
alt="Telegram"></a>
      </div>
      <img src="{% static 'images/logo.png' %}" alt="Bluestock"
class="footer-logo">
      <p>Bluestock Fintech<br>Pune, Maharashtra</p>
      <p>MSME Registration No:<br>UDYAM-MH-01-0138001</p>
      <img src="{% static 'images/logo_6.png' %}" alt="#StartupIndia"
class="startup-img">
    </div>

    <!-- Right section: disclaimer and info -->
    <div class="footer-right">
      <p>
```

```
        Investment in securities markets are subject to market risks, read
all the related documents carefully before investing.
        The users can write to <a
href="mailto:help@bluestock.in">help@bluestock.in</a> for any app, website
related queries.
        Also you can send technical issues to <a
href="mailto:cto@bluestock.in">cto@bluestock.in</a>.
      </p>
      <p class="disclaimer">
        <strong>Disclaimer:</strong> We are not a SEBI registered research
analyst company. We do not provide any kind of stock recommendations,
        buy/sell stock tips, or investment and trading advice. All the
stock snapshots shown on Bluestock app, website, and social media are for
educational purposes only.
      </p>
      <p>
        Before making any investment in the financial market, it is
advisable to consult with your financial advisor.
        Remember that stock markets are subject to market risks.
      </p>
    </div>
  </div>
</div>


  <div class="footer-bottom">
    <p>© Bluestock Fintech All Rights Reserved.</p>
    <p>Made with 🤍 in Pune, Maharashtra</p>
  </div>
</footer>





</body>
</html>
```

## admin_panel/dashboard.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
```

```html
<head>
  <meta charset="UTF-8" />
  <title>Admin Dashboard - Bluestock</title>
  <link rel="stylesheet" href="{% static 'admin_panel/css/admin.css' %}">

  <!-- Optional: Add a little style for the top nav -->
  <style>
    .top-nav {
      display: flex;
      justify-content: space-between;
      align-items: center;
      padding: 10px 20px;
      background: #f8f9fa;
      border-bottom: 1px solid #ddd;
    }

    .top-nav .search-box input {
      padding: 8px 12px;
      border: 1px solid #ccc;
      border-radius: 4px;
    }

    .top-nav .auth-buttons a {
      margin-left: 10px;
      padding: 8px 14px;
      border-radius: 4px;
      text-decoration: none;
      font-size: 14px;
    }

    .top-nav .auth-buttons .login-btn {
      background-color: #5F0EFF;
      color: white;
    }

    .top-nav .auth-buttons .signup-btn {
      background-color: #fff;
      border: 1px solid #5F0EFF;
      color: #5F0EFF;
    }

    .top-nav .auth-buttons span {
      font-weight: 500;
      color: #333;
    }
```

```
  </style>
</head>
<body>

<!-- Sidebar -->
<div class="sidebar">
  <div class="sidebar-header">
    <img src="{% static 'images/logo_7.png' %}" class="logo"
alt="Bluestock FinTech" style="width: 100%; height: auto;">
    <h3>Bluestock Fintech</h3>
  </div>

  <div class="sidebar-title">MENU</div>
  <ul>
    <li><a href="{% url 'admin_panel:dashboard' %}">Dashboard</a></li>
    <li><a href="{% url 'admin_panel:admin_upcoming_ipos' %}">Register
IPO</a></li>
    <li><a href="{% url 'admin_panel:admin_register_ipo' %}">Manage
IPO</a></li>
    <li><a href="{% url 'admin_panel:admin_users' %}">Users</a></li>
    <li><a href="{% url 'admin_panel:home' %}">View Site</a></li>
    {% if user.is_authenticated %}
      <li><a href="{% url 'admin_panel:logout' %}" class="btn btn-
secondary">Logout</a></li>
    {% endif %}
  </ul>

  <div class="sidebar-title">OTHERS</div>
  <ul>
    <li><a href="#">Settings</a></li>
    <li><a href="#">API Manager</a></li>
    <li><a href="#">Accounts</a></li>
    <li><a href="#">Help</a></li>
  </ul>
</div>

<!-- Main Content -->
<div class="main-content">

  <!-- TOP NAV (SEARCH + LOGIN/SIGNUP or User Greeting) -->
  <div class="top-nav">
    <div class="search-box">
      <input type="text" placeholder="Search...">
    </div>
    <div class="auth-buttons">
```

```
      {% if user.is_authenticated %}
        <span>Hi, {{ user.username }}</span>
        <a href="{% url 'admin_panel:logout' %}" class="signup-
btn">Logout</a>
      {% else %}
        <a href="{% url 'admin_panel:login' %}" class="login-
btn">Login</a>
        <a href="{% url 'admin_panel:signup' %}" class="signup-btn">Sign
Up</a>
      {% endif %}
    </div>
  </div>

  <!-- Main Header -->
  <h1>Admin Dashboard</h1>

  <!-- Dashboard Cards -->
  <div class="dashboard-cards">
    <div class="card">Total IPOs: 120</div>
    <div class="card">Upcoming: 15</div>
    <div class="card">Users: 2000</div>
  </div>

  {% block content %}
  <!-- Additional content will be injected here -->
  {% endblock %}

</div>

</body>
</html>
```

## ipo_app/admin.py

```
from django.contrib import admin
from .models import IPO

@admin.register(IPO)
class IPOAdmin(admin.ModelAdmin):
    list_display = ('company_name', 'status', 'open_date', 'listing_date')
```

## admin_panel/admin.py

```
from django.contrib import admin
from .models import IPO

@admin.register(IPO)
class IPOAdmin(admin.ModelAdmin):
    list_display = ('company_name', 'status', 'open_date', 'close_date',
'listing_date')
    search_fields = ('company_name',)
    list_filter = ('status', 'issue_type')
```

## bluestock_project/settings.py

```
import os
from pathlib import Path

BASE_DIR = Path(__file__).resolve().parent.parent

SECRET_KEY = 'your-secret-key'

DEBUG = True

ALLOWED_HOSTS = []

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'rest_framework.authtoken',
    'ipo_app',
    'admin_panel',
    'corsheaders',
    # your other apps
    'social_django',
]


AUTHENTICATION_BACKENDS = (
```

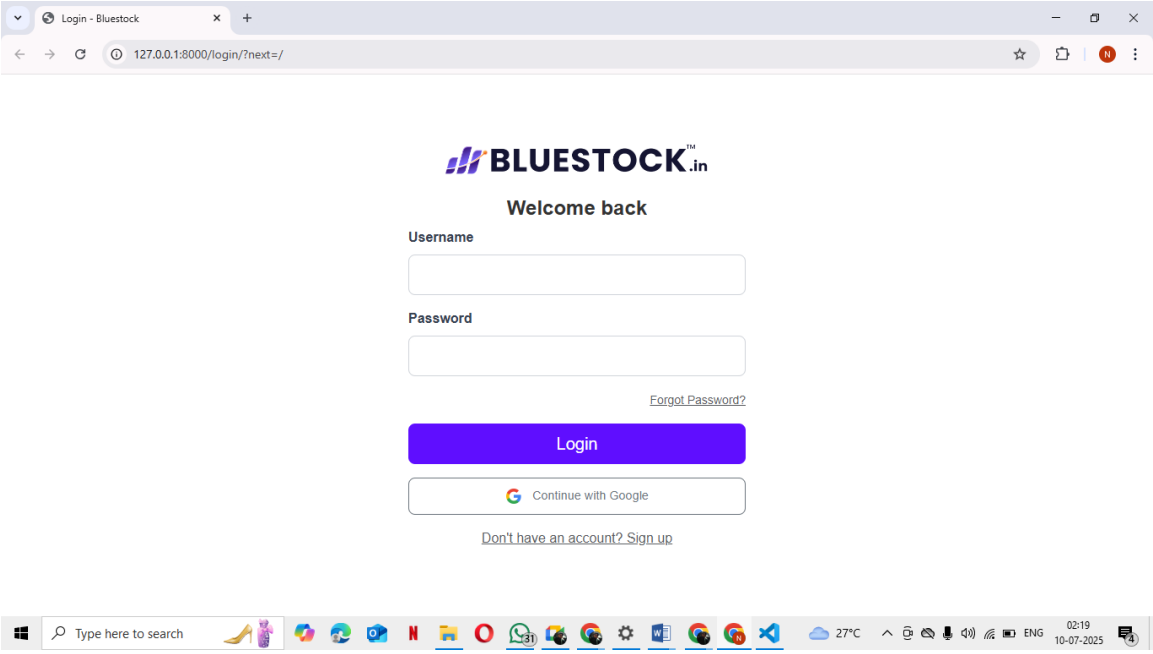
```
    'social_core.backends.google.GoogleOAuth2',  # Google backend
    'django.contrib.auth.backends.ModelBackend',
)

RECAPTCHA_SECRET_KEY = '6LdTpGcrAAAAAE7ViNhFSwkSu4Obdo-wNVVzWc8w'

# Add these keys from Google Developer Console!
SOCIAL_AUTH_GOOGLE_OAUTH2_KEY = '301068552521-
fcf6t132rploqt3nddgpau4sr7ept0v0.apps.googleusercontent.com'
SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET = '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'

LOGIN_URL = '/login/'           # Unauthenticated users go here
LOGIN_REDIRECT_URL = '/'        # After login, go to home page
LOGOUT_REDIRECT_URL = '/login/' # After logout, go to login page

EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'

MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'media'  # or os.path.join(BASE_DIR, 'media')

DEBUG = True


REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework.authentication.TokenAuthentication',
        'rest_framework.authentication.SessionAuthentication',
    ),
    'DEFAULT_PERMISSION_CLASSES': (
        'rest_framework.permissions.AllowAny',
    )
}

REST_FRAMEWORK = {
    'DEFAULT_PAGINATION_CLASS':
'rest_framework.pagination.PageNumberPagination',
    'PAGE_SIZE': 6
}

MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
```

```
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'social_django.middleware.SocialAuthExceptionMiddleware',

]

ROOT_URLCONF = 'bluestock_project.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
                'social_django.context_processors.backends',
                'social_django.context_processors.login_redirect',
            ],
        },
    },
]

WSGI_APPLICATION = 'bluestock_project.wsgi.application'


DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'bluestock_db',         # your DB name
        'USER': 'bluestock_user',       # your DB user
        'PASSWORD': 'Nil12345',     # your DB password
        'HOST': '127.0.0.1',
        'PORT': '5432',                 # default postgres port
    }
}

CORS_ALLOW_ALL_ORIGINS = True
CORS_ALLOW_ALL_ORIGINS = False
CORS_ALLOWED_ORIGINS = [
    'http://localhost:8000',
```

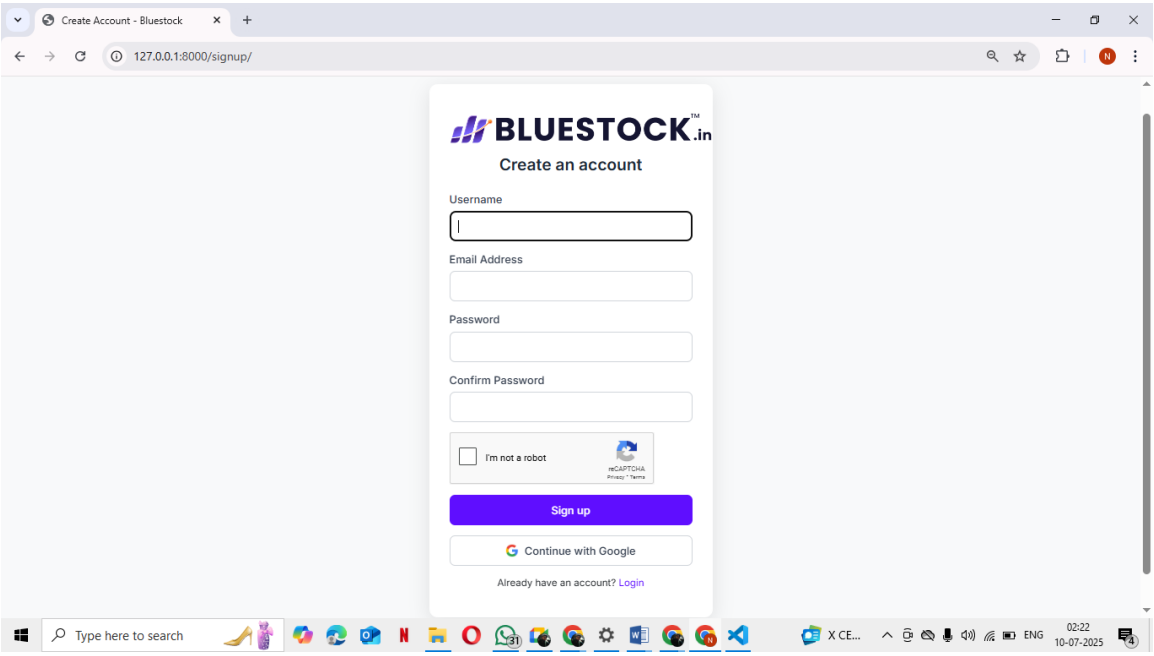
```
    'http://127.0.0.1:8000',
]



STATIC_URL = '/static/'
STATICFILES_DIRS = [
    BASE_DIR / 'static',
]

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

# 8.WEBSITE SNAPSHOTS

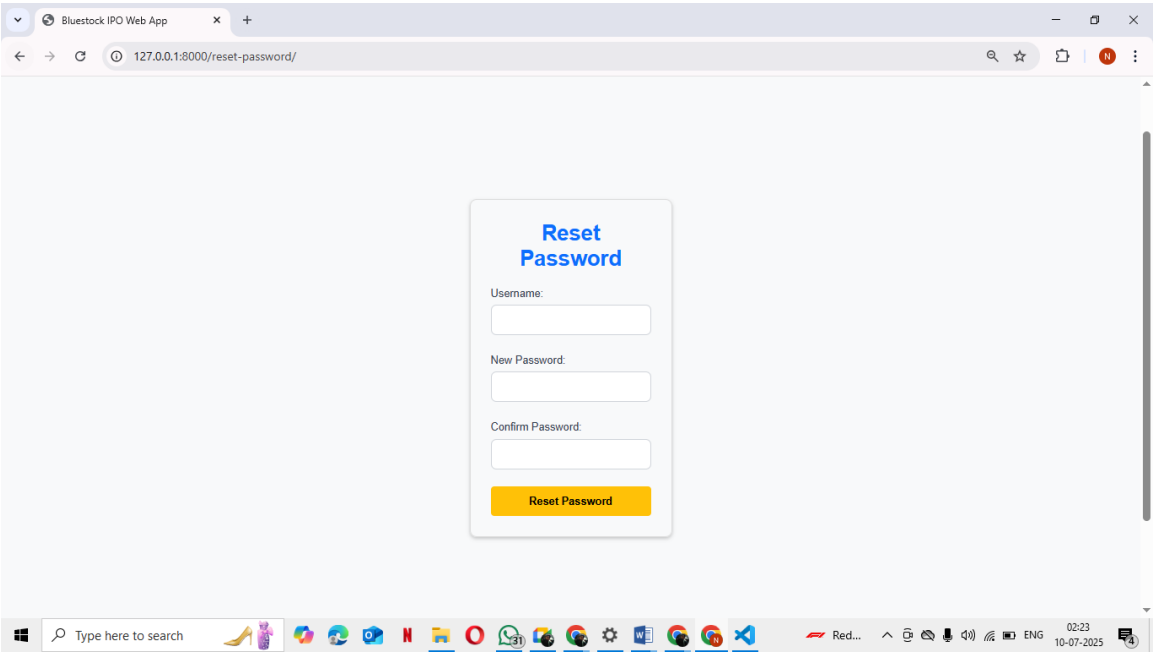
# Login Page



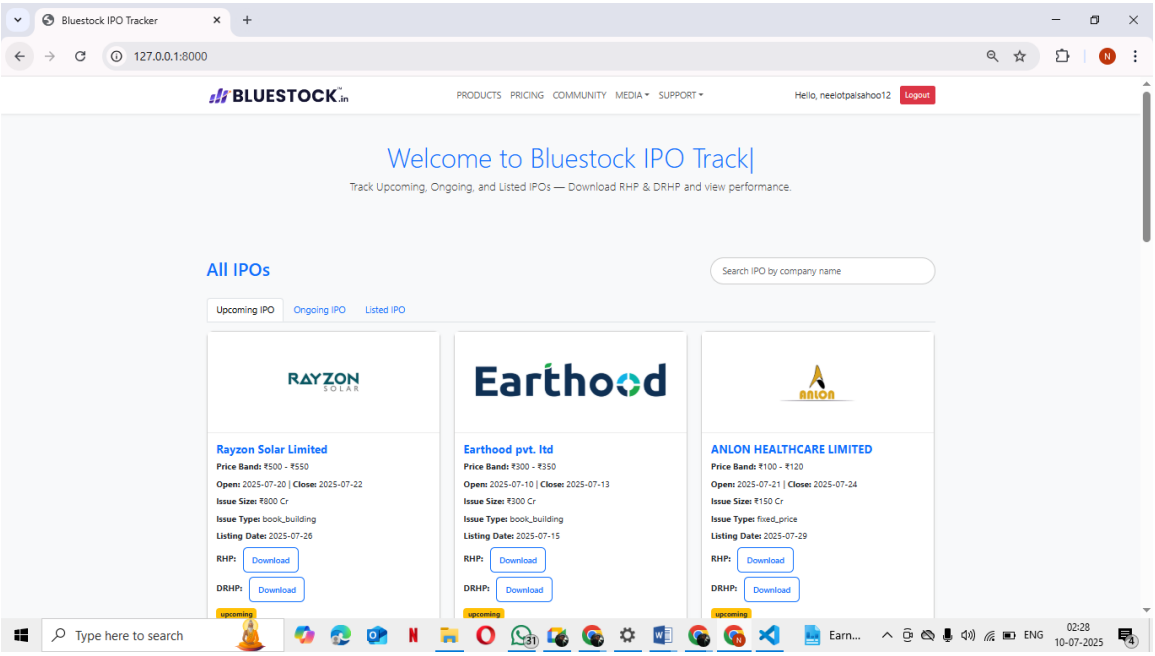

# Signup Page



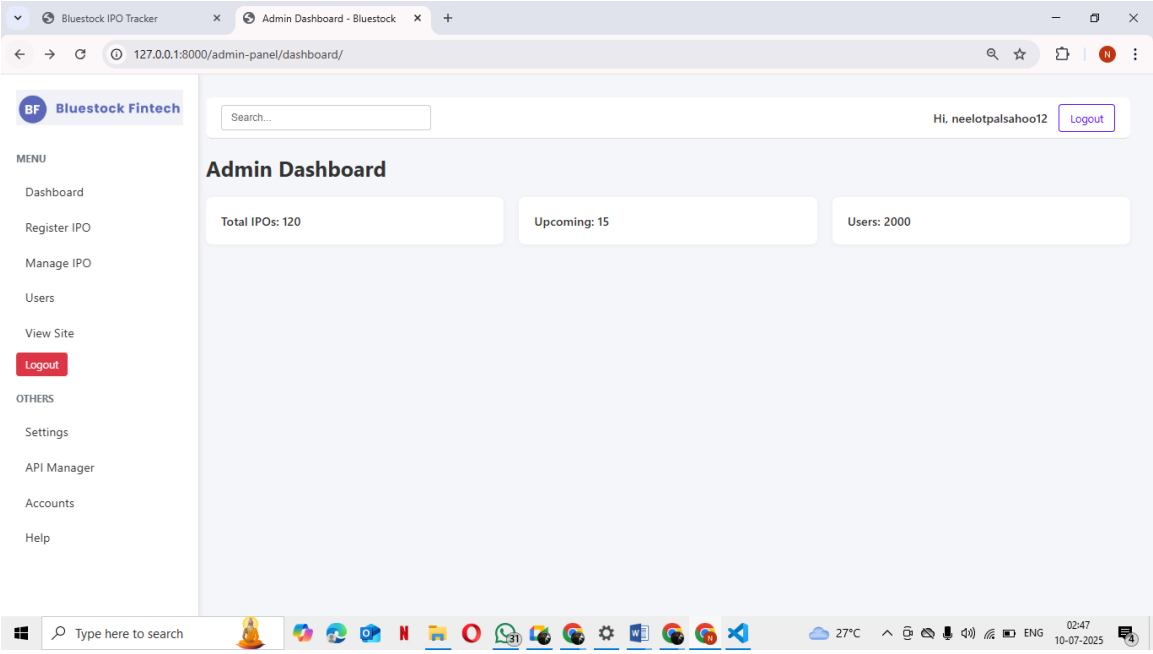
# Reset Password Page



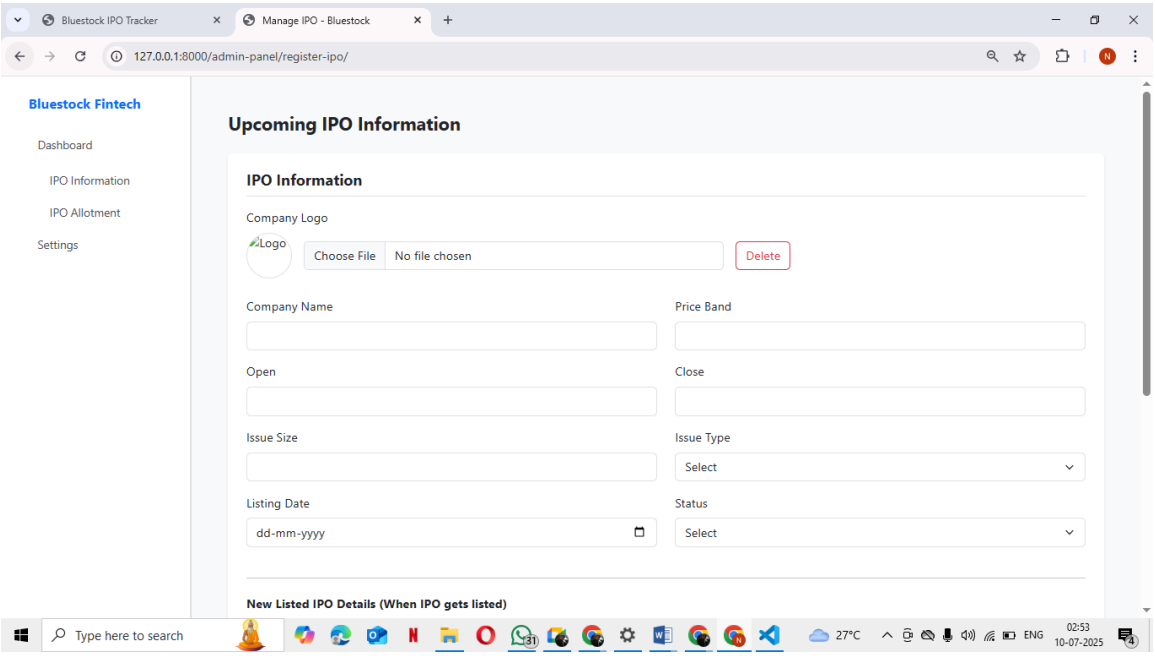

# Ipo_App_frontend Page



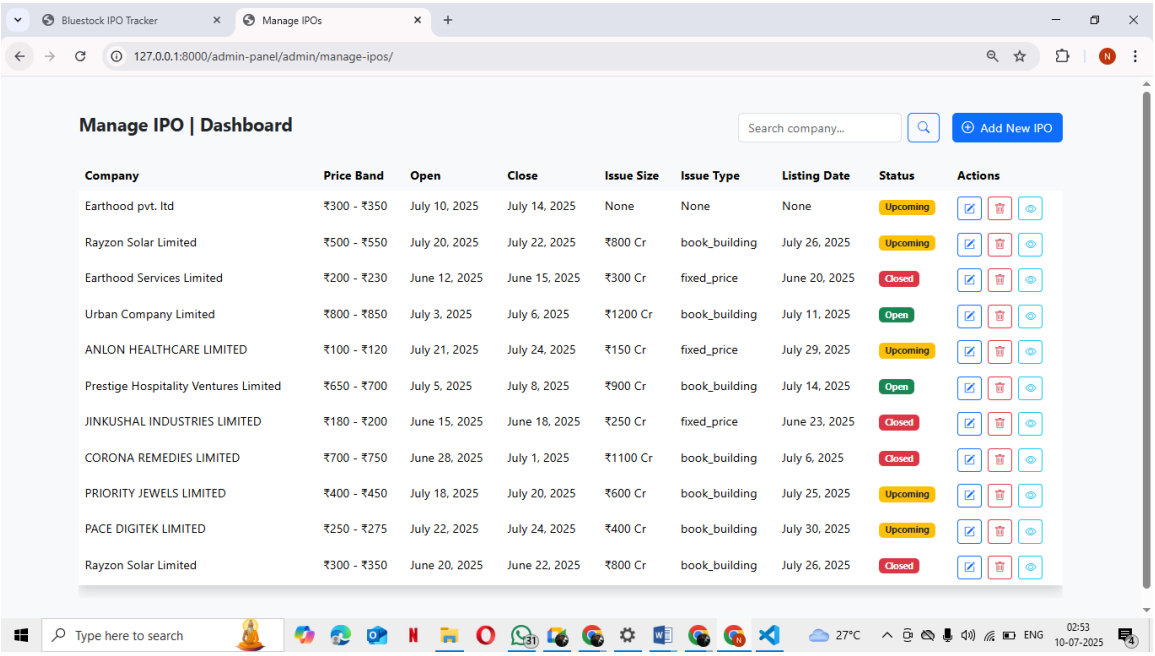
# Admin Dashboard Page



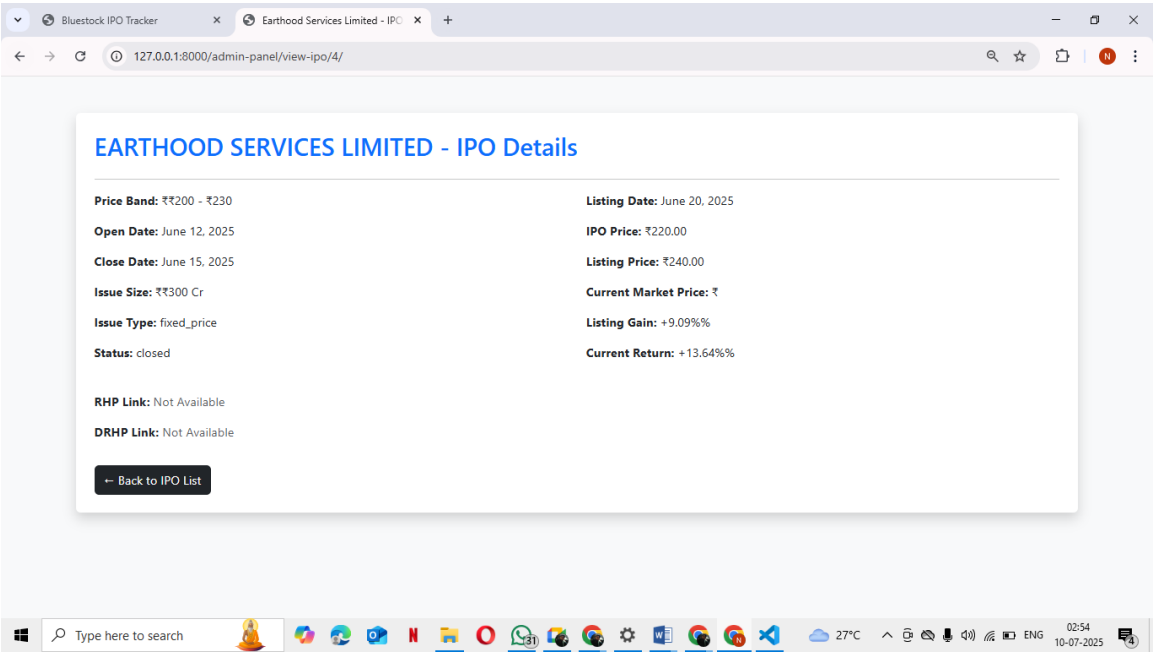

# Register ipo Page



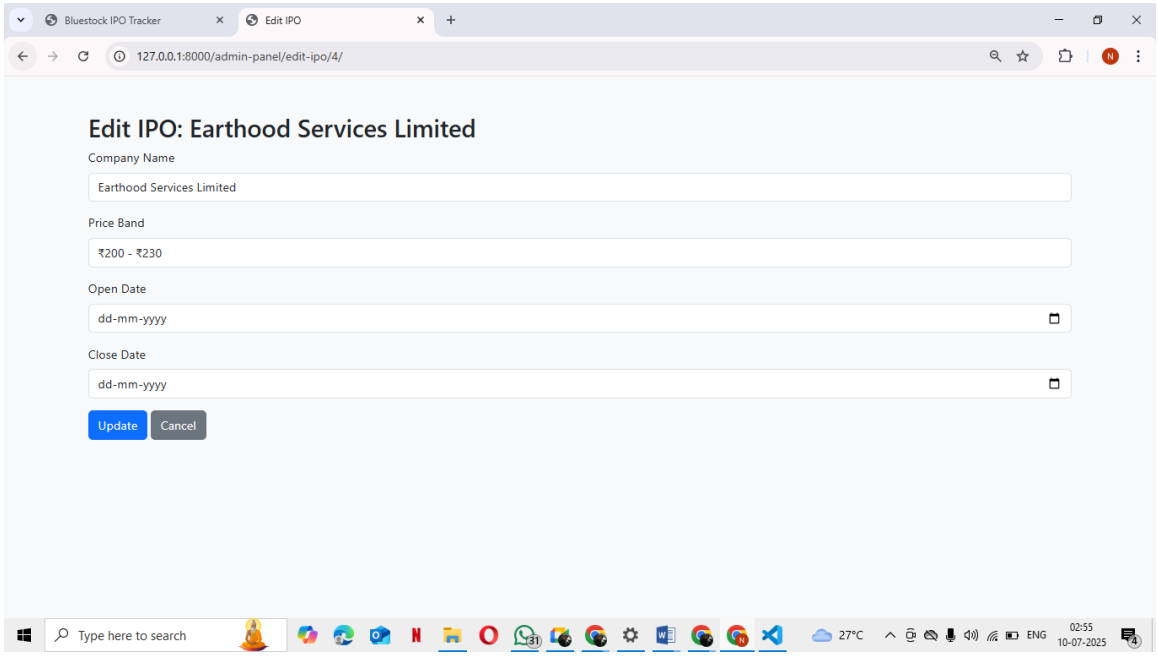
# Manage Ipo Page



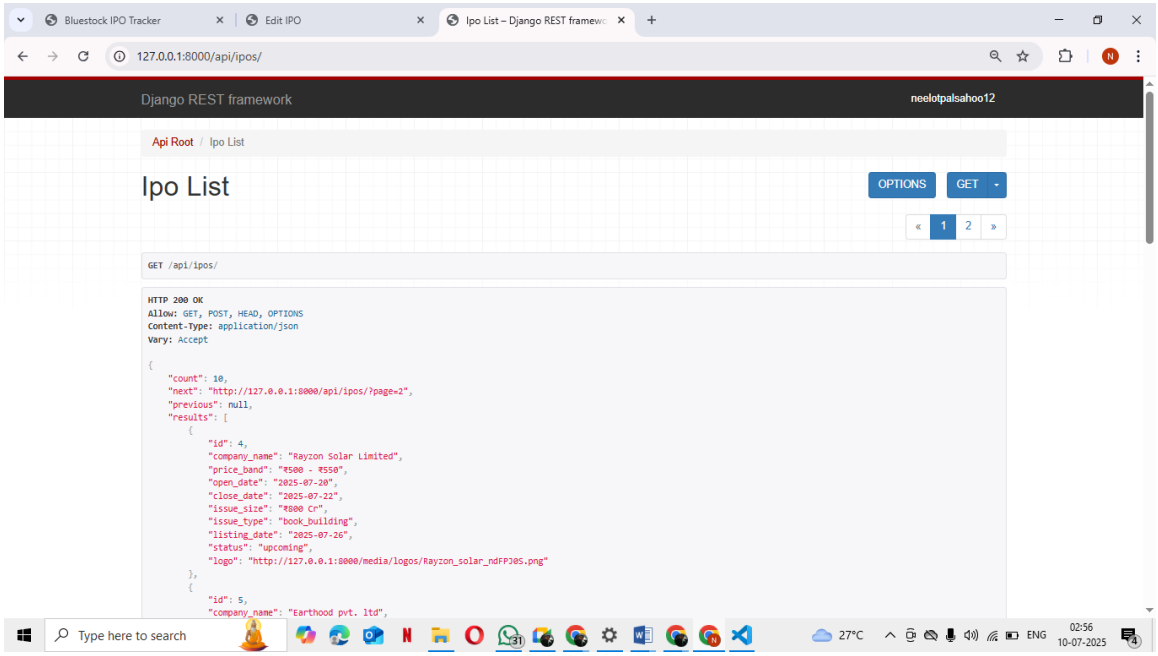

# Ipo details Page



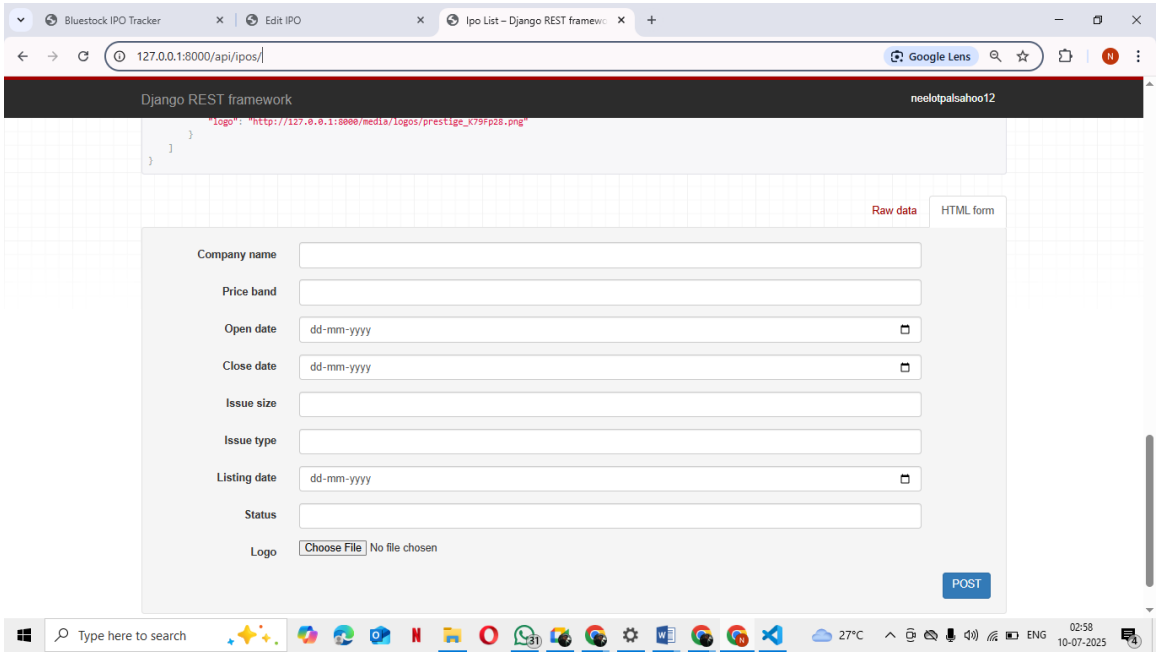
# Edit Ipo Page



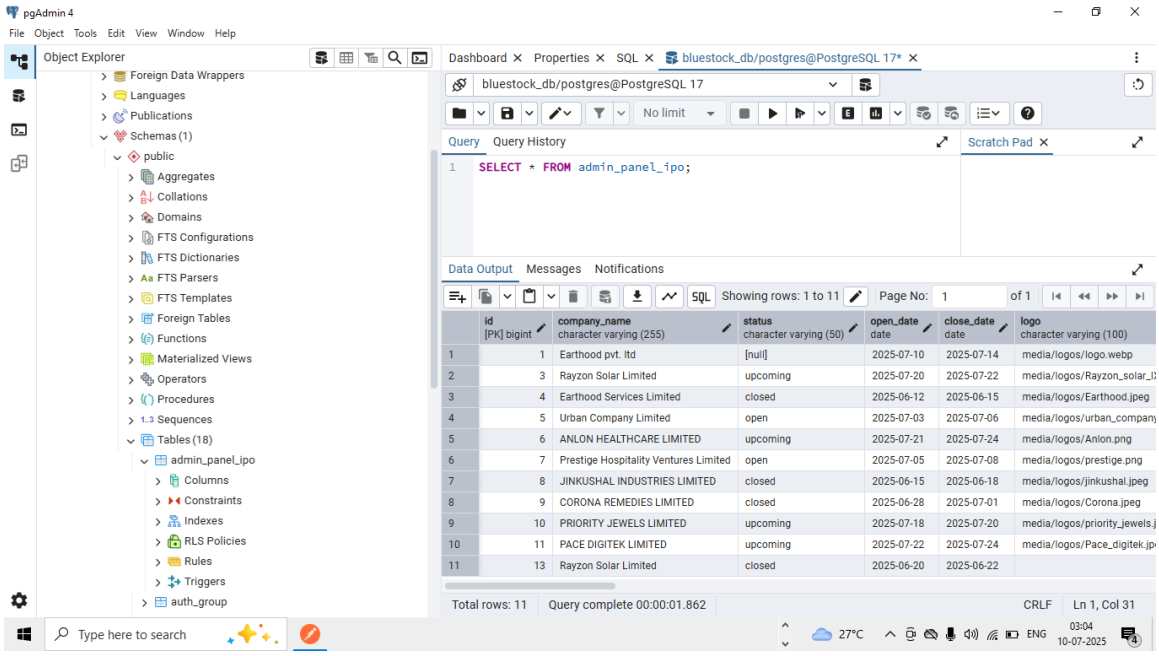

# Ipo list(Api/ipos) Page



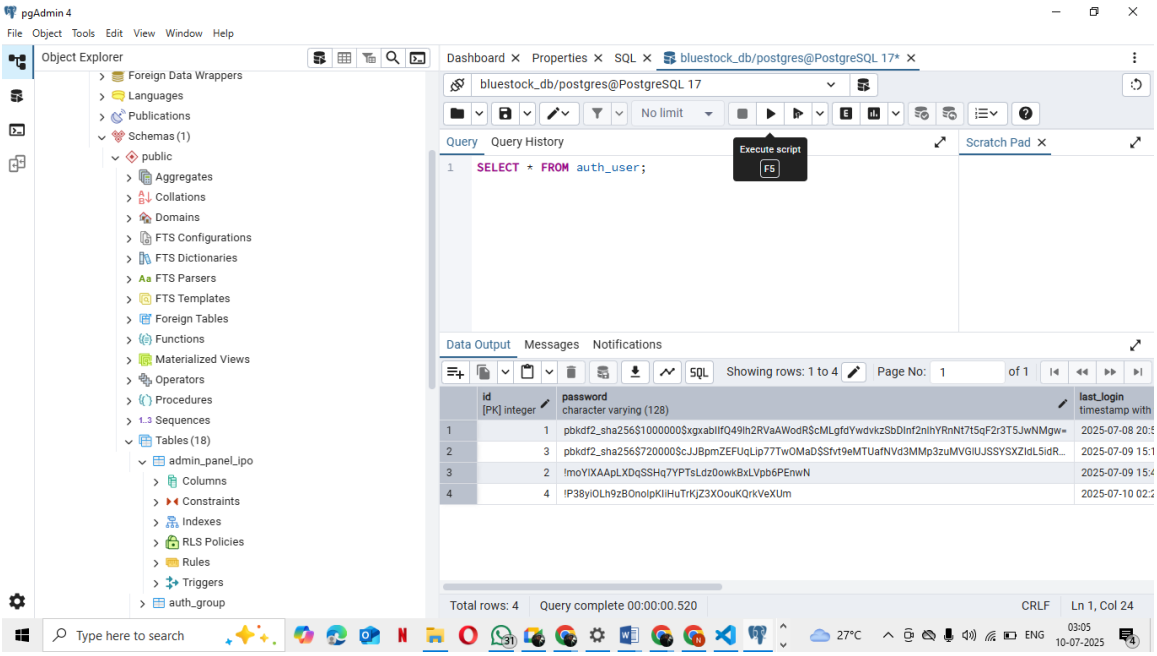
# Ipo Form Page



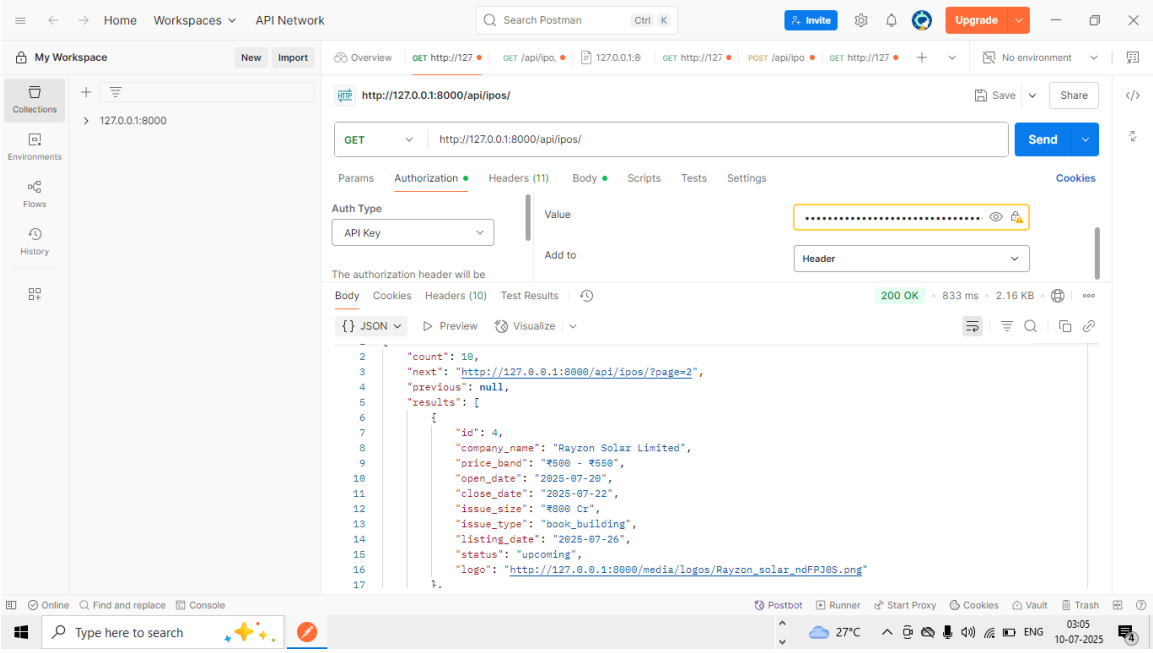

# Database Page For Admin Panel Ipo(PgAdmin4)



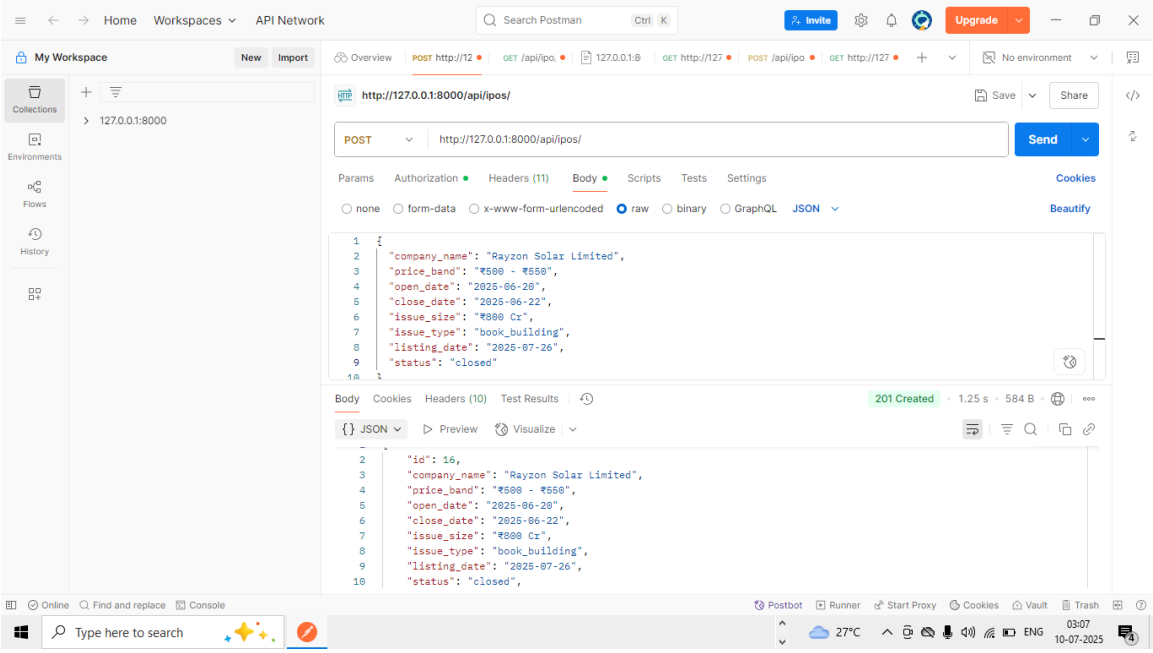
# Database Page For Auth_User(PgAdmin4)



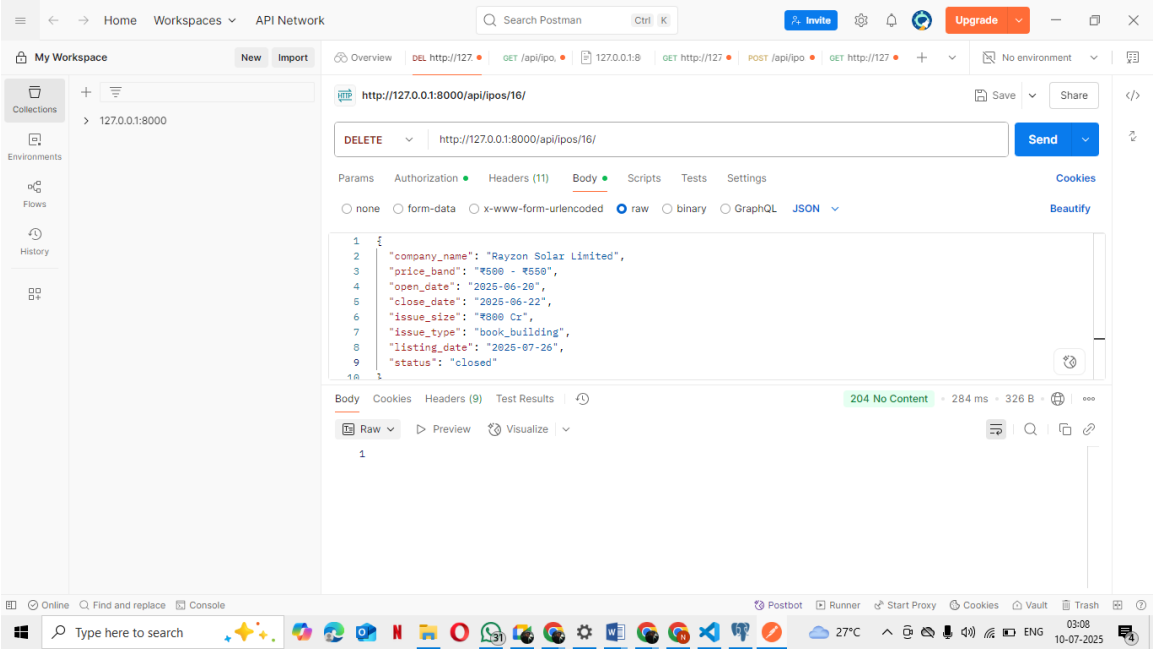
# Postman Testing Using GET for api/Ipos Page



# Postman Testing Using POST for api/Ipos Page



# Postman Testing Using DELETE for api/Ipos Page



# 9. Key Features and User Flow

The Bluestock Fintech platform is designed for simplicity, clarity, and high usability across both user-facing and admin-facing components. Below is a comprehensive breakdown of the key features, interface elements, and user experience flow—from homepage discovery to detailed IPO analysis and backend management.

## Homepage: IPO Discovery & Search

The homepage serves as the primary interface for users to explore upcoming, open, and closed IPOs.

**Features:**

**Tabbed Navigation:**

Tabs categorize IPOs by state:

Upcoming

Open

Closed

Dynamic filtering handled via JavaScript (AJAX or Fetch).

**Search Functionality:**

Live search by company name, status, or issue type.

Implemented with JavaScript fetch() that hits the API and updates DOM in real-time.

**IPO Cards/List:**

Displays essential IPO details:

Company name and logo

Issue size and type

Status and listing date

Buttons to view details or download documents

## Detail Page: IPO Deep Dive

Each IPO has a dedicated detail view that presents complete information and associated files.

**Features:**

**Collapsible Sections** (via Bootstrap accordion or similar):

Company Overview

Pricing & Dates

Gain & Return Analysis

Documents Section (RHP, DRHP)

**Download Links:**

Direct links to:

Red Herring Prospectus (RHP)

Draft RHP (DRHP)

Handled via media/ URLs (secured in production)

**Visual Enhancements:**

Icons, badges, and tooltips enhance UX

Responsive layout optimized for mobile devices

## Admin Dashboard

The admin dashboard is built using Django Admin and enhanced with custom configurations.

**Features:**

**Full CRUD** for:

Companies

IPOs (with gain/return auto-calculated on save)

Documents (PDF upload via file picker)

**Media Management:**

Logos and PDFs uploaded through the interface

Inline previews or download links

**Filters and Search:**

Admin filters by company, status, date

Search fields for name, type, or UUID

**Permissions:**

Only staff/superusers can access the admin panel

Enforced using IsAdminUser for API endpoints

## API Integration Sample

All data shown on the frontend is backed by a secure and structured REST API. Here's a simple integration snippet using JavaScript Fetch API:

**Base Endpoint:** /api/ipos/

**Supports:**

search: by company name or type

filter: by status, issue_type

ordering: by gain, return, issue size

This modular API allows seamless integration with JS frontends, mobile apps, or third-party analytics dashboards.

**User Flow Summary**

Visitor arrives on Homepage

↓

Browses IPO tabs or searches for a company

↓

Clicks on an IPO card to view details

↓

Expands collapsible sections to learn more

↓

Downloads RHP/DRHP documents if needed

↓

(Admins only) Log into dashboard to add/update IPOs and docs

# 10. Development Process & Best Practices

The development of the Bluestock Fintech platform followed structured, iterative, and quality-driven practices to ensure a maintainable, scalable, and secure application. The team adopted an Agile-inspired workflow, reinforced with modern tooling, code quality standards, and collaborative development habits.

## Agile Workflow

The team operated under a lightweight Agile model, focusing on short delivery cycles and regular feedback loops.

**Sprint Planning:**

Development tasks were broken down into 2-week sprints.

Each sprint began with a planning session to define objectives and story estimates.

**Daily Standups:**

15-minute daily standups were held to share:

Progress updates

Blockers or bugs

Plans for the day

Promoted team alignment and quick issue resolution.

**Sprint Reviews & Retrospectives:**

End-of-sprint reviews allowed demos of completed features.

Retrospectives captured what worked, what didn't, and what could be improved.

## Code Quality Enforcement

Maintaining a clean and consistent codebase was a top priority, achieved through automated checks and team conventions.

**Pre-commit Hooks:**

Git pre-commit hooks were configured using pre-commit to auto-run:

flake8: For PEP8 style and linting errors

black: For code auto-formatting

Prevented poorly formatted or non-compliant code from being committed.

## Peer Reviews & Version Control

**Git & GitHub Flow:**

All features and fixes were developed in separate branches.

Followed naming conventions: feature/<name>, bugfix/<issue>

**Pull Requests (PRs):**

Every change was submitted via a PR.

PRs required:

At least one peer review

Passing pre-commit checks

Descriptive title and summary

**Code Review Focus:**

Functionality and correctness

Readability and maintainability

Security and edge cases

## Inline Documentation & Developer Support

**Code Comments:**

Clear inline comments explaining:

Business logic

Edge-case handling

Complex querysets or serializers

## **README.md:**

A detailed README.md was maintained with:

Project setup instructions

Tech stack overview

API usage guides

Contribution workflow

# 11. Timeline & Milestones

The Bluestock Fintech project followed a 5-week development timeline, organized into weekly milestones to deliver a functional, secure, and polished platform. Each week focused on a specific layer of the system—moving from backend foundations to frontend experience and final deployment preparation.

## Week 1: Data Modeling & Migrations

**Milestone:** Project scaffold + Database schema ready

Defined Django models: Company, IPO, and Document

Implemented field constraints, foreign keys, and computed fields (gain, return_percent)

Created and applied initial database migrations

Configured PostgreSQL and tested database connectivity

Verified admin model registration

**Deliverable:** Functional schema with relational integrity

## Week 2: API Layer & Admin Dashboard

**Milestone:** Backend logic and secure APIs implemented

Developed DRF serializers (including nested relationships)

Implemented ModelViewSet for CRUD operations

Added custom validators (e.g., date checks)

Secured API endpoints using IsAdminUser

Configured Django Admin with file upload support and field auto-calculation

**Deliverable:** Working, protected API and admin portal

## Week 3: Frontend Integration & AJAX

**Milestone:** Interactive IPO UI built with real-time data

Built homepage with Bootstrap and tab-based IPO filtering (Upcoming/Open/Closed)

Developed detail pages with collapsible sections and media links

Implemented AJAX-based search and filtering using JavaScript fetch()

Integrated API endpoints with frontend templates

Ensured mobile responsiveness

**Deliverable:** User-friendly UI with real-time interactions

## Week 4: Testing & Documentation

**Milestone:** Quality assurance and dev support

Wrote unit tests for models, serializers, and views

Created integration tests using Postman for all API flows

Conducted cross-browser testing (Chrome, Firefox, Edge)

Added pre-commit hooks (flake8, black) for code quality

Drafted README, API usage notes, and dev setup instructions

**Deliverable:** Tested and documented codebase

## Week 5: UI Polish & Deployment Prep

**Milestone:** Final refinements and readiness for launch

Polished UI elements (buttons, cards, spacing, icons)

Added tooltips, spinners, and confirmation prompts

Verified media handling and file validation in production mode

Finalized deployment checklist (static/media config, production DB, environment vars)

Prepared GitHub repo for handover/demo

**Deliverable:** Production-ready system

# 12. Challenges & Learnings

Throughout the development of the Bluestock Fintech platform, several technical and design challenges emerged across backend, frontend, and infrastructure layers. Each obstacle provided an opportunity to deepen our understanding of Django, REST API design, ORM optimization, and responsive UI development.

## 1. Multi-Field Validation in Serializers

**Challenge:**

Ensuring that date fields (e.g., issue open/close) respected logical ordering and weren't in the past.

Validating business logic such as computed gains based on interdependent fields (issue_price, listing_price).

**Solution:**

Implemented validate() methods in DRF serializers for cross-field logic.

Encapsulated calculations in model save() to ensure database-level consistency.

**Learning:**

DRF's serializer-level validation is powerful but must be combined carefully with model-level logic for safety and reusability.

## 2. Media Storage Configuration

**Challenge:**

Managing user-uploaded media files (logos, RHP/DRHP PDFs) in a clean and secure way.

Handling file path generation and media URL accessibility in both development and production.

**Solution:**

Configured MEDIA_ROOT and MEDIA_URL in Django settings.

Used structured upload paths (logos/, rhps/, drhps/) for clarity.

Added file size/type validators to prevent misuse.

**Learning:**

Django handles file storage elegantly, but production deployments require additional layers (e.g., S3, NGINX, or WhiteNoise).

## 3. ORM Performance Tuning

**Challenge:**

Nested serializers and related object queries (e.g., IPO → Company → Documents) caused unnecessary database hits (N+1 problem).

**Solution:**

Used select_related() and prefetch_related() in ViewSets to optimize joins and reduce query count.

**Learning:**

Profiling Django queries early prevents performance bottlenecks in list views and APIs.

## 4. Responsive Design Edge Cases

**Challenge:**

Collapsible sections, long document names, and stacked cards broke layout on small screens.

**Solution:**

Leveraged Bootstrap grid and utilities like text-truncate, overflow-wrap, and accordion patterns.

Tested using Chrome device emulator and real mobile devices.

**Learning:**

Mobile-first design must account for variable content lengths, tap targets, and touch-friendly interactions.

## 13. Conclusion

The development of the Bluestock Fintech platform was a comprehensive, hands-on journey through the full software development lifecycle. It provided an immersive experience across all layers of a modern web application—backend modeling, API design, frontend integration, file handling, testing, and deployment preparation.

Through iterative sprints, rigorous validation, and consistent code quality practices, the team successfully delivered a robust and secure IPO management system that meets all functional and technical requirements.

Key achievements include:

A clean and normalized database schema with automated financial computations

A RESTful API layer supporting CRUD, search, filtering, and secure access

A responsive, AJAX-powered frontend UI with downloadable IPO documents

A production-ready admin dashboard for managing companies, IPOs, and media

Strong adherence to best practices in version control, testing, and documentation

The project also offered valuable lessons in:

Real-world validation logic

Media storage design

ORM query optimization

Cross-browser and mobile-first UI design

Ultimately, this project showcases a real-world, full-stack application built from the ground up with modern development principles and a user-centered design approach.

## 14. References

1. Django Documentation

2. Django REST Framework Guide

3. Bootstrap 5 Documentation

4. Bluestock Figma Designs

5. Bluestock HR Guidelines