Section One: Introduction to Python (6 hours)

Hour 1: Introduction to Python Programming Language, Installation & Setup

- **Objective**: Understand Python, its applications, and get set up.
- **Topics**:
  - Overview of Python (history, key features, use cases)
  - Setting up Python: Installation (using Python.org or Anaconda)
  - Installing IDE (VS Code or PyCharm)
  - Running a simple Python script
  - Introduction to Python shell
- **Doubt-solving session**: Address installation issues, confusion on environment setup.

Hour 2: Basic Python Syntax, Variables, Data Types & Operators

- **Objective**: Learn basic syntax, variables, and data types.
- **Topics**:
  - Variables and assignment
  - Data types: Integers, Strings, Lists, Tuples, Dictionaries, Booleans
  - Type conversion
  - Operators: Arithmetic, Comparison, Logical, and Assignment operators
- **Hands-on**: Create a small program to calculate the area of a circle, working with user input.
- **Doubt-solving session**: Clarify variable assignment, data types, and operators with examples.

Hour 3: Control Flow Statements and Functions

- **Objective**: Understand how to control the flow of code using conditions and loops.
- **Topics**:
  - Conditional statements (if-else)
  - Loops (for, while)
  - Functions and function arguments/return values
  - Exception handling (try-except)
- **Hands-on**: Write a program that calculates the factorial of a number using both recursion and iteration.
- **Doubt-solving session**: Debug and clarify logic errors in loops and conditionals.

Hour 4: Introduction to Object-Oriented Programming (OOP)

- **Objective**: Learn the basics of OOP principles in Python.

- **Topics**:
    - Classes and objects
    - Attributes and methods
    - Inheritance, polymorphism, and encapsulation
    - Creating simple classes
- **Hands-on**: Create a class representing a `Car` with attributes like model, make, and methods for starting and stopping the car.
- **Doubt-solving session**: Review OOP concepts, common mistakes when defining classes and methods.

Section Two: Web Development Basics (7 hours)

Hour 5: Introduction to Web Development, HTML & CSS

- **Objective**: Basic understanding of web development, HTML, and CSS.
- **Topics**:
    - Web application fundamentals: Client-server architecture
    - Introduction to HTML tags, elements, attributes
    - Basic styling with CSS (selectors, properties, layouts)
- **Hands-on**: Create a simple webpage with an HTML form and CSS styling.
- **Doubt-solving session**: Troubleshoot common issues with HTML structure and CSS formatting.

Hour 6: Setting Up Local Development Environment (VS Code) & Version Control (Git)

- **Objective**: Get comfortable with the development environment and version control.
- **Topics**:
    - Setting up VS Code (extensions for Python, Django)
    - Installing Git and basic Git commands (clone, commit, push, pull)
    - Using GitHub for code hosting
- **Hands-on**: Initialize a local Git repository, commit changes, and push to GitHub.
- **Doubt-solving session**: Resolve issues with Git setup and pushing to GitHub.

Hour 7: Introduction to Django Framework, Installation & Project Setup

- **Objective**: Learn the basics of the Django web framework.
- **Topics**:
    - Installing Django using `pip` and setting up a new project
    - Overview of Django structure (settings, apps, models, views, URLs)
    - Running the Django development server

- **Hands-on**: Create a new Django project and start the server to check if everything works.
- **Doubt-solving session**: Discuss any errors with installation, Django server, and project setup.

Hour 8: Creating a Basic Django Web Application (Views, Templates, URLs)

- **Objective**: Build the skeleton of a basic Django web app.
- **Topics**:
  - Creating Django apps within a project
  - Views, Templates, and URL routing
  - Serving dynamic content using Django views and templates
  - Use of `render()` function in Django views
- **Hands-on**: Create a "Hello World" page and a contact form that handles user input.
- **Doubt-solving session**: Clarify issues related to URL routing, views, and templates.

Section Three: Django Rest Framework (DRF) (9-11 hours)

Hour 9: Introduction to Django Rest Framework (DRF) and Its Features

- **Objective**: Learn about DRF and how it simplifies building APIs.
- **Topics**:
  - What is Django Rest Framework (DRF)?
  - DRF components: Serializers, Views, ViewSets, Routers
  - Why use DRF? (ease of creating RESTful APIs)
  - Installing DRF and setting up a simple API
- **Hands-on**: Install DRF, create a simple API to retrieve and create data in a model.
- **Doubt-solving session**: Go over the DRF installation and initial configuration.

Hour 10: Building a RESTful API with DRF (Serializers, Views, URL Routing)

- **Objective**: Build a basic REST API using DRF for a sample model.
- **Topics**:
  - Serializers in DRF and how they convert complex data types (e.g., models) into JSON
  - Creating API views (function-based and class-based)
  - Setting up URL routing for API endpoints
- **Hands-on**: Application development as of technical specification
- **Doubt-solving session**: Address issues with serializers and API views, discuss common pitfalls.

Hour 11: Advanced DRF Features: Authentication & Permissions

- **Objective**: Understand how to manage authentication and permissions for APIs.
- **Topics**:
  - Authentication in DRF: Token-based authentication, Session Authentication
  - Permissions: Controlling access to API endpoints based on roles
  - Writing custom permissions
- **Hands-on**: Add token-based authentication and create custom permissions for an API endpoint.
- **Doubt-solving session**: Debug authentication errors, clarify permission management.

Hour 12: Advanced DRF Features: Pagination & Filtering

- **Objective**: Learn how to paginate and filter API data for more efficient data handling.
- **Topics**:
  - DRF pagination and how to paginate large data sets
  - Adding filtering capabilities to API endpoints (query parameters)
  - Using `django-filter` for advanced filtering
- **Hands-on**: Implement pagination and filtering in application
- **Doubt-solving session**: Help resolve issues with paginated responses and filtering logic.

Hour 13: DRF Testing: Writing Unit Tests for APIs

- **Objective**: Learn how to write unit tests for DRF APIs to ensure stability and correctness.
- **Topics**:
  - Introduction to testing in Django using `unittest` and DRF's `APIClient`
  - Writing tests for DRF views and serializers
  - Running tests and interpreting results
- **Hands-on**: Writing test cases as per requirement.
- **Doubt-solving session**: Discuss common mistakes in writing and running tests.

Section Four: Working with Frontend & Additional DRF Features (3-5 hours)

Hour 14: Integrating Frontend with DRF (Optional)

- **Objective**: Learn how to integrate DRF with a frontend (HTML/JavaScript).
- **Topics**:
  - Making AJAX requests to DRF API from a frontend

    - Using fetch API or Axios to connect frontend with backend
    - Displaying API data in HTML pages
- **Hands-on**: Fetch data from the DRF API and display it on a webpage.
- **Doubt-solving session**: Resolve issues related to AJAX requests, API responses, and CORS.

Hour 15: Introduction to Django Admin for Managing Data

- **Objective**: Learn how to use Django's built-in admin interface to manage your database.
- **Topics**:
    - Setting up and customizing Django admin
    - Registering models with admin site
    - Customizing list views and form layouts
- **Hands-on**: Django Admin Dashboard
- **Doubt-solving session**: Clarify issues with admin customization and model registration.

Hour 16: Deploying the Django App Locally & Troubleshooting

- **Objective**: Learn how to deploy a Django app locally on your machine and troubleshoot common deployment issues.
- **Topics**:
    - Local deployment steps (collect static files, configure settings)
    - Using SQLite vs PostgreSQL for local development
    - Troubleshooting common deployment issues (database connection, static files not loading)
- **Hands-on**: Deploy the Django app locally on your computer with all configurations and troubleshoot any issues.
- **Doubt-solving session**: Help resolve deployment issues and clarify any confusion about local setup.

Final Hour: Review, Q&A, and Final Project

Technical Specifications:

1. Relational Database:

- **Database:** SQLite (Django's default database)
- **Tables:** Users, Artists, Songs
- **CRUD Operations:** Handled via Django ORM for managing the records of the users, artists, and songs.

2. Language and Framework:

- **Programming Language:** Python
- **Framework:** Django (Web framework for building the application)
- **API Framework:** Django Rest Framework (For building RESTful APIs for handling CRUD operations)

Core Features:

a. Initial Landing Page:

1. **Login Screen:**
    - Admin users should land on a login page.
    - **New Registration:** Admin users can register a new account. After registration, they will be redirected to the login screen.
    - **Login:** Once the admin logs in, they will be directed to the dashboard page.
2. **Redirection Logic:**
    - If the admin is already logged in, they should be redirected directly to the dashboard page.

b. Dashboard Page:

- After logging in, the admin will be directed to the **Dashboard Page**.
- On this page, the admin can perform **CRUD operations** for the tables (Users, Artists, Songs).
- A **Logout button** will be available to log out from the session.

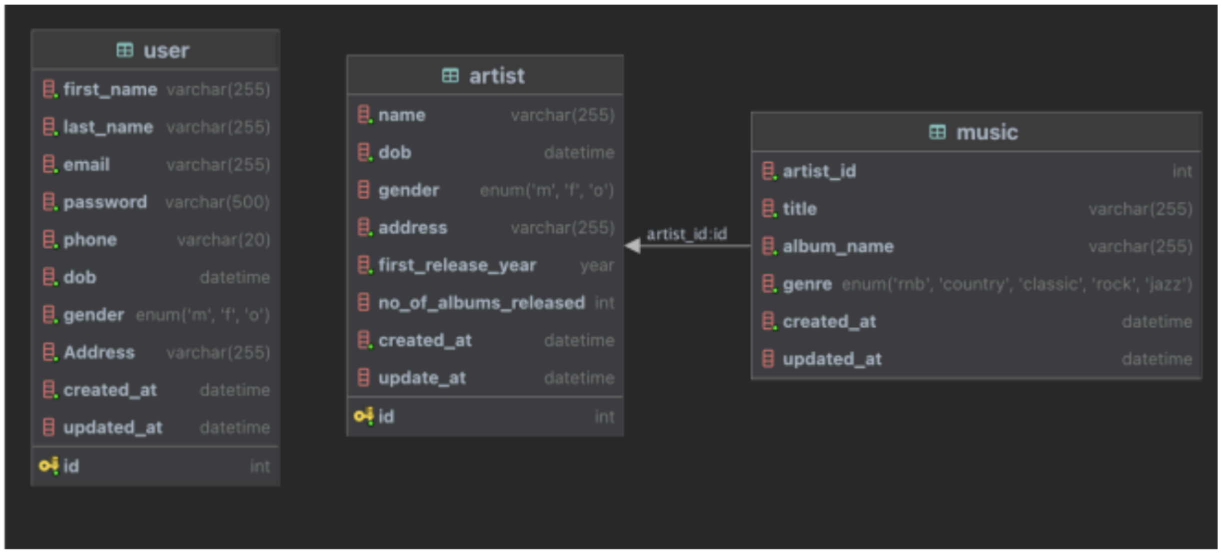
c. Users Page:

- Admin can:
    - **List** all users’ records.
    - **Create a new user** (with necessary fields like name, email, password, etc.).
    - **Update** the information of an existing user.
    - **Delete** a user record.

d. Artists Page:

- Admin can:
    - **List** all artists.
    - **Create a new artist** with details like name, genre, etc.
    - **Update** the information of an existing artist.
    - **Delete** an artist.
    - **Create Songs** for each artist (attach songs to an artist's profile).
    - **List songs** associated with the artist.
    - **Update songs** of the artist.
    - **Delete songs** of the artist.

Schema Diagram for Database


user
first_name varchar(255)
last_name varchar(255)
email varchar(255)
password varchar(500)
phone varchar(20)
dob datetime
gender enum('m', 'f', 'o')
Address varchar(255)
created_at datetime
updated_at datetime
id int
artist
name varchar(255)
dob datetime
gender enum('m', 'f', 'o')
address varchar(255)
first_release_year year
no_of_albums_released int
created_at datetime
update_at datetime
id int
artist_id:id
music
artist_id int
title varchar(255)
album_name varchar(255)
genre enum('rnb', 'country', 'classic', 'rock', 'jazz')
created_at datetime
updated_at datetime