Full-Stack Project Training Program: Python, Django, and React JS

Prerequisites

- Basic knowledge of Python (variables, loops, functions), HTML, and CSS.
- Python 3.8+ installed.
- Node.js and npm installed for React.
- A code editor (e.g., VS Code).
- Git and GitHub account for version control.
- Terminal or command prompt access.

Project Overview

Task Management Application:

- **Tech Stack**:
  - Backend: Python, Django, Django REST Framework, SQLite (development), PostgreSQL (production).
  - Frontend: React JS, Tailwind CSS, Axios.
  - Tools: Git, Postman, Vercel (frontend), Render (backend).

Program Structure

- **Days 1-8**: Python Basics
- **Days 9-19**: Django Basics and Django API
- **Days 20-30**: React JS
- **Days 31-40**: Final-Year Project Development

Python Basics

Day 1: Python Setup and Basics

- **Topics**:
  - Install Python and set up a virtual environment.
  - Variables, data types (int, str, list, dict), and basic operations.
  - String formatting and input/output.
- **Tasks**:
  - Write a program to calculate the area of a triangle.
  - Create a simple program to greet users based on input.
- **Resources**: Python official documentation, W3Schools Python.
- **Time**: 5 hours.

Day 2: Control Structures

- **Topics**:
  - Conditional statements (if, elif, else).
  - Loops (for, while).
  - Break and continue statements.
- **Tasks**:
  - Write a program to determine if a year is a leap year.
  - Print a multiplication table using loops.
- **Time**: 5 hours.

Day 3: Functions

- **Topics**:
  - Defining functions, default arguments, and return values.
  - Lambda functions.
  - Scope and recursion.
- **Tasks**:
  - Write a function to check if a string is a palindrome.
  - Create a recursive function to calculate Fibonacci numbers.
- **Time**: 5 hours.

Day 4: Modules and Packages

- **Topics**:
  - Importing standard modules (e.g., math, random).
  - Creating and using custom modules.
  - Installing packages with pip.
- **Tasks**:
  - Create a module with utility functions (e.g., string manipulation).
  - Use the `random` module to build a dice-rolling program.
- **Time**: 5 hours.

Day 5: Data Structures - Lists and Tuples

- **Topics**:
  - Lists: methods, slicing, and comprehensions.
  - Tuples: immutability and unpacking.
- **Tasks**:
  - Write a program to sort a list of numbers.
  - Use list comprehension to filter odd numbers.
- **Time**: 5 hours.

Day 6: Data Structures - Dictionaries and Sets

- **Topics**:

    - Dictionaries: key-value pairs, methods, and iteration.
    - Sets: operations (union, intersection).
- **Tasks**:
    - Create a program to count word frequencies in a text.
    - Use sets to find common elements in two lists.
- **Time**: 5 hours.

Day 7: File Handling

- **Topics**:
    - Reading and writing text files.
    - Working with CSV files.
    - Using `with` statement for file operations.
- **Tasks**:
    - Write a program to log tasks to a text file.
    - Read a CSV file and display its contents.
- **Time**: 5 hours.

Day 8: Exception Handling and Review

- **Topics**:
    - Exception handling (try, except, finally).
    - Custom exceptions.
    - Review Python basics.
- **Tasks**:
    - Handle errors in a file-reading program.
    - Build a CLI to-do list app storing tasks in a file.
- **Milestone**: Complete a simple Python CLI to-do list application.
- **Time**: 5 hours.

Django Basics and Django API

Day 9: Django Introduction and Setup

- **Topics**:
    - Install Django and create a project (`mysite`).
    - Understand Django's MVT architecture.
    - Configure settings and run the server.
- **Tasks**:
    - Set up a virtual environment and install Django.
    - Create a Django project and access the default page.
- **Resources**: Django official documentation.
- **Time**: 4 hours.

Day 10: Django App and Views

- **Topics**:
  - Create an app (`pages`).
  - Create function-based views for homepage and about page.
  - Configure URLs for the app.
- **Tasks**:
  - Create a `pages` app with a homepage view.
  - Map URLs for homepage and about page.
- **Time**: 4 hours.

Day 11: Templates and Static Files

- **Topics**:
  - Create templates (`base.html`, `home.html`, `about.html`).
  - Use template inheritance.
  - Serve static files (CSS, images).
- **Tasks**:
  - Create a base template with navigation.
  - Add a logo image and basic CSS styling.
- **Time**: 4 hours.

Day 12: Models and Admin

- **Topics**:
  - Define a `Task` model (title, description, created_at, image).
  - Create and apply migrations.
  - Set up the Django admin interface.
- **Tasks**:
  - Create a `Task` model and register it in admin.
  - Add sample tasks via the admin interface.
- **Time**: 4 hours.

Day 13: Dynamic Content

- **Topics**:
  - Pass data from views to templates using context dictionaries.
  - Display dynamic content in templates.
- **Tasks**:
  - Update the homepage to display a list of tasks.
  - Use template filters for date formatting.
- **Time**: 4 hours.

Day 14: Forms and Image Uploads

- **Topics**:
  - Create a `ModelForm` for the `Task` model.
  - Configure media files for image uploads.

    - Handle form submissions.
- **Tasks**:
    - Create a form to add tasks with image uploads.
    - Display task images on the homepage.
- **Time**: 4 hours.

Day 15: User Authentication

- **Topics**:
    - Use Django's authentication system.
    - Implement login, logout, and registration.
    - Restrict task creation to logged-in users.
- **Tasks**:
    - Add user registration and login forms.
    - Use `@login_required` for task creation.
- **Time**: 4 hours.

Day 16: Introduction to Django REST Framework

- **Topics**:
    - Install Django REST Framework (DRF).
    - Understand REST principles and HTTP methods.
    - Create an `api` app.
- **Tasks**:
    - Install DRF and configure settings.
    - Create an `api` app.
- **Resources**: DRF official documentation.
- **Time**: 4 hours.

Day 17: Serializers and API Views

- **Topics**:
    - Create serializers for the `Task` model.
    - Build API views using DRF's generic views.
- **Tasks**:
    - Create a `TaskSerializer`.
    - Build a `TaskListCreateAPIView` and test with Postman.
- **Time**: 4 hours.

Day 18: CRUD API Endpoints

- **Topics**:
    - Implement CRUD endpoints using DRF ViewSets.
    - Configure API URLs.
- **Tasks**:
    - Create a `TaskViewSet` for CRUD operations.

    - Test all endpoints with Postman.
- **Time**: 4 hours.

Day 19: API Authentication and CORS

- **Topics**:
    - Implement token-based authentication.
    - Configure CORS for frontend integration.
- **Tasks**:
    - Set up DRF token authentication.
    - Install `django-cors-headers` and test.
- **Milestone**: Complete a Django backend with a REST API for task management.
- **Time**: 4 hours.

React JS

Day 20: React Setup and Basics

- **Topics**:
    - Install Node.js and create a React app with Vite.
    - Understand JSX and components.
    - Set up Tailwind CSS.
- **Tasks**:
    - Create a React app (`task-manager-frontend`).
    - Build a `Header` component with Tailwind styling.
- **Resources**: React documentation, Tailwind CSS documentation.
- **Time**: 5 hours.

Day 21: Components and Props

- **Topics**:
    - Create reusable components.
    - Pass props to components.
- **Tasks**:
    - Create a `TaskCard` component for task display.
    - Pass dummy task data as props.
- **Time**: 5 hours.

Day 22: State and Hooks

- **Topics**:
    - Use `useState` and `useEffect` hooks.
    - Manage component state.
- **Tasks**:
    - Build a toggle component using `useState`.

    - Fetch dummy data with `useEffect`.
- **Time**: 5 hours.

Day 23: React Router

- **Topics**:
    - Set up React Router for navigation.
    - Create routes for home, about, and task form.
- **Tasks**:
    - Implement navigation with `Link` and `Route`.
    - Set up basic routing.
- **Time**: 5 hours.

Day 24: Fetching Data

- **Topics**:
    - Use Axios to fetch data from Django API.
    - Display API data in components.
- **Tasks**:
    - Fetch tasks from the Django API.
    - Render tasks using `TaskCard`.
- **Time**: 5 hours.

Day 25: Forms in React

- **Topics**:
    - Create controlled forms.
    - Handle form submissions to APIs.
- **Tasks**:
    - Build a task creation form.
    - Submit form data to the Django API.
- **Time**: 5 hours.

Day 26: File Uploads

- **Topics**:
    - Handle file uploads in React.
    - Send files to Django API.
- **Tasks**:
    - Add an image upload field to the task form.
    - Test image uploads via the API.
- **Time**: 5 hours.

Day 27: Authentication

- **Topics**:

    - Implement login/logout functionality.
    - Store tokens in local storage.
- **Tasks**:
    - Create login and registration forms.
    - Integrate with Django API authentication.
- **Time**: 5 hours.

Day 28: State Management

- **Topics**:
    - Use Context API for global state.
    - Manage user and task state.
- **Tasks**:
    - Create a context for authentication.
    - Share task data across components.
- **Time**: 5 hours.

Day 29: Styling and Responsiveness

- **Topics**:
    - Polish UI with Tailwind CSS.
    - Ensure responsive design.
- **Tasks**:
    - Style task list and forms.
    - Test responsiveness on mobile and desktop.
- **Time**: 5 hours.

Day 30: Testing React

- **Topics**:
    - Write basic tests for components.
    - Use React Testing Library.
- **Tasks**:
    - Test the `TaskCard` component.
    - Test form submission.
- **Milestone**: Complete a React frontend for the task manager.
- **Time**: 5 hours.

Project Development

Day 31: Project Planning

- **Topics**:
    - Define project scope and requirements.
    - Set up Git for version control.

- **Tasks**:
  - Write a project specification for the Task Management Application.
  - Initialize a Git repository.
- **Time**: 5 hours.

Day 32: Backend Setup

- **Topics**:
  - Finalize Django project structure.
  - Configure database and media files.
- **Tasks**:
  - Set up PostgreSQL for production.
  - Ensure media file configuration.
- **Time**: 5 hours.

Day 33: Backend API Development

- **Topics**:
  - Complete API endpoints for tasks and authentication.
  - Test API functionality.
- **Tasks**:
  - Finalize CRUD endpoints.
  - Test with Postman.
- **Time**: 5 hours.

Day 34: Frontend Setup

- **Topics**:
  - Finalize React project structure.
  - Set up routing and components.
- **Tasks**:
  - Complete routing for all pages.
  - Ensure component reusability.
- **Time**: 5 hours.

Day 35: Frontend Integration

- **Topics**:
  - Connect React frontend to Django API.
  - Implement all features (task CRUD, authentication).
- **Tasks**:
  - Complete task listing and form submission.
  - Test authentication flow.
- **Time**: 5 hours.

Day 36: File Upload Integration

- **Topics**:
  - Ensure image uploads work seamlessly.
  - Display images in the frontend.
- **Tasks**:
  - Test image uploads end-to-end.
  - Verify image display.
- **Time**: 5 hours.

Day 37: Testing

- **Topics**:
  - Write tests for backend and frontend.
  - Perform end-to-end testing.
- **Tasks**:
  - Test task creation and image uploads.
  - Write unit tests for key components.
- **Time**: 5 hours.

Day 38: Debugging and Optimization

- **Topics**:
  - Debug issues in the application.
  - Optimize API calls and frontend performance.
- **Tasks**:
  - Fix CORS or authentication issues.
  - Reduce API response times.
- **Time**: 5 hours.

Day 39: Deployment

- **Topics**:
  - Deploy backend to Render.
  - Deploy frontend to Vercel.
- **Tasks**:
  - Configure environment variables.
  - Test deployed application.
- **Time**: 5 hours.

Day 40: Documentation and Presentation

- **Topics**:
  - Document the project.
  - Prepare a presentation.
- **Tasks**:
  - Write a project report.
  - Create a slide deck and demo the project.

- **Milestone**: Complete and deploy the Task Management Application.
- **Time**: 5 hours.

Final Project Structure

```
django_basics/
    manage.py
    mysite/
        settings.py
        urls.py
        ...
    pages/
        models.py
        views.py
        urls.py
        forms.py
        templates/
            base.html
            home.html
            about.html
            task_form.html
        static/
            styles.css
            logo.png
    api/
        serializers.py
        views.py
        urls.py
    media/
        tasks/
    task-manager-frontend/
        src/
            components/
                TaskCard.jsx
                Header.jsx
            App.jsx
            index.css
        package.json
```