BACKEND DEVELOPMENT WITH PYTHON AND DJANGO CURRICULUM

The **Backend Development with Python and Django** curriculum is a year-long program designed to help you develop the core skills needed for building web applications and RESTful APIs using Python and Django. Throughout this course, you will progress from the basics of Python to advanced Django concepts, working on real-world projects, and gaining hands-on experience with backend development, database management, user authentication, and deployment.

By the end of this curriculum, you will:

- Be proficient in Python programming and Django framework.
- Understand database management, security, and deployment strategies.
- Have practical experience developing RESTful APIs and working with databases.
- Complete a capstone project to showcase your skills.

This curriculum is ideal for both beginners and intermediate learners.

Course Structure:

- **Sessions per Week:** 2
- **Duration:** 1 Year
- **Phases:** 4
- **Level:** Beginner to Intermediate

Phase 1: Introduction to Python and Django Fundamentals

Duration: 12 Weeks
Focus: Basic Python programming, Django setup, templates, basic models, Git version control.

Week 1-3: Python Basics

- Variables, Data Types, Input/Output
- Conditional Statements, Loops
- Lists, Tuples, Dictionaries, and Sets
- **Error Handling**: try/except blocks
- **Functions**: Creating and using functions
- **Project:** Build a simple text-based "Quiz App" to test users' general knowledge.

Week 4-6: Introduction to Django

- Setting up Django, Project, and App Creation
- Model-View-Template (MVT) Architecture
- Defining Models and Admin Interface
- **Django Settings and Structure**: Understanding settings, apps, and structure
- **Project:** Build a "To-Do List" app where users can create, view, and delete tasks.

Week 7-9: Django Views and Templates

- Rendering HTML templates, Static files, Extending templates
- Basic Class-Based Views (CBVs)
- **Django Forms**: Introduction to form handling with validation
- **Project:** Extend the "To-Do List" app with user authentication (login/logout) functionality.

Week 10-12: Version Control and Deployment

- **Git Basics**: Cloning, Commit, Push, Branching
- **Basic Deployment on Vercel**: Deploying Django apps with PostgreSQL integration
- **Managing Environment Variables**: Using tools like django-environ for settings
- **Project:** Deploy the "To-Do List" app to Vercel with PostgreSQL integration.

Phase 1 Project:

- **Website:** Build a simple portfolio website with Django, showcasing the projects you've done in this phase (including your department, level, hostel, number, GitHub link). Deploy it to a platform like Vercel.

Phase 2: Intermediate Django Concepts

Duration: 12 Weeks
Focus: Intermediate Django concepts, user authentication, database management, and testing.

Week 1-3: Advanced Django Models

- Relational Databases, Foreign Keys, Many-to-Many relationships
- Managing Models in Django Admin
- **Database Indexing and Optimization**: Techniques for improving database performance
- **Project:** Build a "Blog App" with models for blog posts, categories, and tags.

Week 4-6: Django Views and URLs

- Handling Dynamic URLs and Template Rendering

- Class-Based Views: ListView, DetailView
- **Project:** Extend the "Blog App" with a feature that allows users to create, update, and delete blog posts.

Week 7-9: User Authentication and Authorization

- Implementing User Login, Logout, and Registration
- Using Django's User Model and Custom User Models
- **Role-based Permissions**: Custom user roles and access control
- **Project:** Build a "User Authentication System" where users can register, log in, and update their profile details.

Week 10-12: Testing and Debugging in Django

- Writing Unit Tests, Mocking Data
- **Integration Testing**: Testing how components work together
- **Debugging and Troubleshooting**: Debugging Django projects effectively
- **Project:** Write unit tests for your "Blog App" and "User Authentication System."

Term 2 Project:

- **Message Board App:** Develop a "Message Board App" where users can post and comment on messages, with user authentication implemented.

Phase 3: Introduction to APIs, Advanced Django, and API Security

Duration: 12 Weeks
Focus: Introduction to RESTful APIs, Django REST Framework (DRF), user authentication for APIs, and API security.

Week 1-3: Introduction to REST APIs with Django REST Framework (DRF)

- Basic RESTful API Concepts, Serializers
- ViewSets, API Views, and Routers
- **API Versioning**: Using DRF versioning for maintaining API versions
- **Project:** Build a "Book Management API" with CRUD operations using Django REST Framework.

Week 4-6: Handling Authentication in APIs

- Using Token Authentication
- Using JWT (JSON Web Tokens) for Authentication
- **OAuth and Third-party Authentication**: Implementing authentication with external providers (e.g., Google, GitHub)

- **Project:** Enhance the "Book Management API" to support JWT authentication for secure access.

Week 7-9: Permissions and Access Control in APIs

- Using Django's Permission System
- Role-based Permissions and Custom Permissions
- **Project:** Add role-based access control (admin and regular user) to your "Message Board App" REST APIs.

Week 10-12: Testing REST APIs

- Writing Tests for API Views, Serializers
- Mocking External API Calls
- **API Documentation**: Using tools like drf-yasg for automatic API documentation
- **Project:** Write tests for your "Book Management API" and document it using Swagger.

Term 3 Project:

- **Blog API:** Develop a "Blog API" using Django REST Framework, where users can create, read, update, and delete blog posts. Implement user authentication with JWT and role-based permissions for API access.

Phase 4: Mastering Development Tools and Deployment

Duration: 12 Weeks
Focus: Mastery of development tools, API documentation, testing, and deployment strategies.

Week 1-3: Exploring Development and Testing Tools

- **Postman**: Setting up API requests and testing REST APIs
- **Swagger/OpenAPI**: Documenting APIs with drf-spectacular and generating Swagger documentation
- **Project:** Document and test the "Blog API" using Postman and Swagger. Generate a comprehensive API documentation page with interactive examples.

Week 4-6: Version Control and Collaboration with GitHub

- **GitHub Basics**: Cloning repositories, managing branches, and creating pull requests
- **Collaborative Workflows**: Using issues, project boards, and GitHub Actions for CI/CD
- **Code Quality**: Implementing linters and code formatters
- **Project:** Collaboratively enhance the "Blog API" with a team on GitHub. Practice code reviews and GitHub Actions for automated testing.

Week 7-9: Deployment Essentials

- Deploying Django Applications to platforms like **Vercel**, **Heroku**, or **AWS**
- **PostgreSQL with Vercel**: Setting up the database for production
- **Cloudinary for Media Storage**: Configuring Cloudinary for handling media files
- **Environment Variables**: Managing .env files securely
- **Project:** Deploy the "Blog API" to Vercel with PostgreSQL integration and Cloudinary for media files.

Week 10-12: Capstone Project - E-commerce Website

- **Capstone Project Overview**: Build a complete E-commerce Website with the following features:
    - **User Authentication**: User registration, login, and profile management with JWT authentication.
    - **Product Management**: CRUD operations for products (name, description, price, images, etc.).
    - **Category Management**: Categorizing products for easy browsing.
    - **Shopping Cart and Orders**: API endpoints for managing shopping cart and orders.
    - **Payment Integration**: Integrating Paystack or Stripe for payment processing.
    - **Deployment**: Host the application on Vercel, with Cloudinary for product images and user-uploaded media.
    - **Testing and Optimization**: Testing the API and writing unit tests for key functionality.
    - **Frontend Integration**: Optionally, integrate the backend with a frontend framework (React/Vue) for a complete web application.

Capstone Submission Requirements:

- A **GitHub repository** with organized code, clear commit history, and a README file documenting the API features and deployment instructions.
- **Deployed live E-commerce Website** with full functionality.
- Detailed **Swagger documentation** link for the API.
- **Demonstration** of the website with a complete shopping process (from login to checkout).

Here are the detailed project requirements for each week of the curriculum, broken down week by week:

Phase 1: Introduction to Python and Django Fundamentals

Week 1-3: Python Basics

Project: Quiz App

- Implement a basic quiz app that:
    - Ask multiple-choice questions.
    - Tracks user answers and calculates a score at the end.
    - Presents results to the user.
    - Use basic Python functions, conditional statements, loops, and data structures (lists, dictionaries).

Week 4-6: Introduction to Django

Project: To-Do List App

- Build a simple to-do list app where:
    - Users can create, view, and delete tasks.
    - Use Django models to store tasks.
    - Display tasks on a webpage using Django templates.

Week 7-9: Django Views and Templates

Project: To-Do List with Authentication

- Extend the To-Do List App to include user authentication:
    - Allow users to register, log in, and log out.
    - Allow users to create, view, and delete their own tasks only.
    - Implement Django's built-in User model for authentication.

Week 10-12: Version Control and Deployment

Project: Portfolio Website

- Create a personal portfolio website that includes:
    - A homepage with an introduction and projects listed.
    - A contact form that users can submit.
    - Deployment to a platform like Vercel or Heroku.
    - Basic responsive design to ensure it looks good on mobile and desktop.
    - Use Git for version control and manage the project through GitHub.

Phase 2: Intermediate Django Concepts

Week 1-3: Advanced Django Models

Project: Blog App

- Create a blog app that:
    - Allows users to add, edit, and delete blog posts.
    - Posts should have categories and tags.
    - Use Django models to store blog posts, categories, and tags.
    - Display posts on the site with options to filter by category or tag.

Week 4-6: Django Views and URLs

Project: Blog App with CRUD Operations

- Extend the Blog App to include:
    - Allow users to create, read, update, and delete blog posts.
    - Implement URL patterns for post detail and editing views.
    - Use Class-Based Views (CBVs) for better structure (e.g., ListView, DetailView).

Week 7-9: User Authentication and Authorization

Project: User Authentication System

- Implement user authentication:
    - Users can register, log in, log out, and manage their profile (name, email, password).
    - Allow users to update their profile details.
    - Use Django's built-in authentication system to handle login/logout.
    - Secure sensitive user data by hashing passwords and ensuring proper user access to only their own data.

Week 10-12: Testing and Debugging in Django

Project: Blog App and User Authentication System Testing

- Write unit tests for:
    - CRUD operations in the Blog App.
    - User authentication functionality (login, registration, profile management).
    - Test views, models, and templates to ensure they work as expected.
    - Handle edge cases such as invalid user input or non-existent posts.

Phase 3: Introduction to APIs, Advanced Django, and API Security

Week 1-3: Introduction to REST APIs with Django REST Framework (DRF)

Project: Book Management API

- Build a REST API for managing books:
    - Implement CRUD operations for books (title, author, description, etc.).
    - Use DRF serializers for data handling.
    - Create API views to manage book records.
    - Implement pagination to limit the number of books returned.

Week 4-6: Authentication in APIs

Project: Enhance Book Management API with JWT

- Integrate JWT (JSON Web Tokens) for user authentication:
    - Secure the API by allowing only authenticated users to access certain endpoints.
    - Implement token-based authentication in the API.
    - Allow users to log in and receive a JWT for future requests.

Week 7-9: Permissions and Access Control in APIs

Project: Message Board API with Role-Based Access

- Create a REST API for a message board:
    - Users can post messages and comment on them.
    - Implement role-based permissions using DRF's permission classes.
    - Restrict certain actions (e.g., delete posts) to admin users.
    - Secure the API with JWT authentication.

Week 10-12: Testing REST APIs

Project: Book Management API Testing

- Write tests for the Book Management API:
    - Test CRUD operations for books.
    - Write tests for authentication and authorization (JWT).
    - Test different scenarios (e.g., missing fields, unauthorized access).
    - Use tools like Postman for testing API endpoints.

Phase 4: Mastering Development Tools and Deployment

Week 1-3: Exploring Development and Testing Tools

Project: API Documentation and Testing with Swagger/OpenAPI

- Document the Book API or Message Board API:

    - Use Swagger/OpenAPI with `drf-spectacular` to auto-generate API documentation.
    - Add descriptions for each endpoint and its parameters.
    - Make the API documentation interactive so users can test endpoints directly from the docs.
    - Test the APIs using Postman collections and ensure they function as expected.

Week 4-6: Version Control and Collaboration with GitHub

Project: Collaborative Blog API Enhancement

- Work with a team to enhance the Blog API:
    - Use GitHub to collaborate, manage branches, and create pull requests.
    - Practice code reviews and resolve merge conflicts.
    - Set up GitHub Actions for continuous integration and automated testing.

Week 7-9: Deployment Essentials

Project: Deploy the Blog API to Vercel

- Deploy the Blog API to a cloud platform (Vercel or Heroku):
    - Set up PostgreSQL as the database.
    - Use Cloudinary for media storage (images for blog posts).
    - Secure environment variables with `.env` files.
    - Test API endpoints in the live environment to ensure functionality.

Week 10-12: Capstone Project - E-commerce Website

Capstone Project: E-commerce Website

- Develop a full-featured e-commerce website(Figma file will be provided):
    - Implement user authentication with JWT for secure login and registration.
    - Create CRUD operations for products (title, description, price, images).
    - Implement a shopping cart and order management system with checkout functionality.
    - Integrate payment processing (Paystack or Stripe).
    - Deploy the website to a cloud platform and use Cloudinary for media storage.
    - Write unit tests for key functionality.
    - Integrate the backend with a frontend (e.g., React, Vue.js) to complete the web app.
    - Generate comprehensive API documentation using Swagger.