COURSE TITLE: Python Backend Development			
Course Code::		**Course Type:**	**IC**
Teaching Hours/ Week (L:T:P):	**0:1:4**	**Credits:**	3
Total Teaching Hours:	**0+15+60**	**CIE + SEE Marks:**	**50+50=100**

Preamble *(brief description).*

This course on Python Backend Development offers a comprehensive introduction to building robust, scalable, and efficient backend systems. By focusing on core principles of backend development, such as server-side programming, API design, database integration, and application security, students will acquire the skills necessary to create dynamic web applications. The course emphasizes the practical application of Python's extensive libraries and frameworks, such as Flask and Django, to develop RESTful APIs, manage data, and ensure seamless communication between the client and server. Through hands-on projects and real-world scenarios, students will gain critical problem-solving skills and the expertise required to architect reliable backend solutions for modern applications.

Course Outcomes

At the end of the course students will be able to…

CO1	Understand the principles and importance of backend development in modern software systems.
CO2	Implement backend functionalities using Python, focusing on frameworks like Flask and Django.
CO3	Design and develop RESTful APIs to facilitate seamless client-server communication.
CO4	Integrate databases with backend applications for efficient data management.
CO5	Analyze and ensure the performance, scalability, and security of backend systems.

PO – CO mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8
CO1	3	3	2	2	1	-	-	2
CO2	3	3	3	2	1	-	-	2
CO3	3	3	3	3	2	-	-	2
CO4	3	3	3	2	1	-	-	2
CO5	3	3	3	2	1	2	2	3

Note:

Course content

Theory

1. Overview of Client-Server Architecture.
2. HTTP Protocol, REST.
3. Web Services.
4. Overview of Backend Frameworks in Python: Flask.
5. Overview of Backend Frameworks in Python: Django.
6. RESTful API Design: Principles and Best Practices.
7. RESTful API Design: Principles and Best Practices.
8. API Documentation and Tools (Postman).
9. API Documentation and Tools (Swagger).
10. CRUD Operations using Flask.
11. CRUD Operations using Django REST Framework (DRF).
12. Introduction to Databases.
13. User Authentication: Tokens, Sessions.
14. User Authentication: OAuth2.
15. Securing APIs: HTTPS, CORS, Rate Limiting.

Practical

Week	**Contents (*Practical)***
1	Create a simple view that returns "Hello, World!" and map it to a URL.
2	Create a view that displays a list of hyperlinks to various social media websites and map it to a URL in urls.py.
3	Create a base template with a navigation bar and footer. Extend this base template in two child templates, one for a homepage and another for an "About Us" page (Controller functionality).
4	Create a model for a blog post with fields: title, content, and published_date. Add, retrieve, modify, and delete records using the Django shell. Configure Django to use a SQLServer database. Migrate an existing model to the SQLServer database. Create a model to store user-uploaded images and display these images on a gallery page.
5	Register the blog post model in the admin panel. Customize the admin interface to display only the title and published date fields.
6	Create a form for a "Contact Us" page with fields for name, email, and message. Handle form submission and display a success message upon form submission.
7	Configure email settings in settings.py. Create a function that sends an email to the site admin when the contact form is submitted.
8	Create a Web API using Django REST Framework to manage blog posts (CRUD operations). Also implement web API using flask.

9	Create a serializer for the blog post model. Use the serializer to handle JSON requests and responses. Create a function-based view to handle API requests for creating and retrieving blog posts.
10	Implement JWT Authentication and Cookie-based Authentication in Django.
11	Implement authorization in Django.
12	Demonstrate full-stack development skills: Choose a problem and implement a solution showcasing the Python backend development techniques.

Suggested Reading:

Text Books:

1. Flask Web Development: Developing Web Applications with Python by Miguel Grinberg.
2. Django for Professionals by William S. Vincent.
3. Python API Development Fundamentals by Jack Chan.

Web Resources

1. Flask Official Documentation
2. Django Official Documentation
3. REST API Design Guidelines
4. Real Python Tutorials
5. Postman Learning Center

Video Resources

1. **Corey Schafer's Python Tutorials** (YouTube).
2. **Traversy Media's Backend Development Tutorials** (YouTube).
3. NPTEL Backend Development Courses.

Part B: Course Evaluation System

Assessment System	Assessment Component	Description	Weightage	Marks
Practical Continuous Internal Evaluation (CIE)	CIE-I	Students are instructed to form a group of not more than 2 students and have to compile a project with the laboratory experiences and present it at the designated dates, scheduled by the department: Marks distribution may be considered as follows (**50 Marks)**: ❖ Synopsis Presentation: 10 % ❖ Presentation 1(After 8 weeks): 30% ❖ Presentation2(After 12 weeks): 40% ❖ Report : 20 % —----------	5 M 15 M 20 M 10 M	50 M
CIE (Practical)			**50%**	**50 M**
Semester End Evaluation (SEE)	**SEE**	**Lab Examination conducted for 2 hours duration****	**50%**	**50 M**
TOTAL MARKS			**100%**	**100**

** Semester End Examinations are conducted with the COE Designated Examiners.

- Marks distributions can be considered as follows:
  - Write Up: 15%
  - Final presentation: 40%
  - Viva voce Examination:30%
  - Report: 15%