Course: BTech **Semester:** 4

Prerequisite: Basic knowledge of Programming and web applications

Rationale: This course provides a broad introduction to Python programming and development of web applications. Developing and using Python as a scripting language for automating tasks and data processing. Moreover Building and deploying web applications using popular Python frameworks such as Django and Flask.

Teaching and Examination Scheme

Teaching Scheme					Examination Scheme					Total
Lecture Hrs/Week	Tutorial Hrs/Week	Lab Hrs/Week	Hrs/Week	Credit	Internal Marks			External Marks		
					T	CE	P	T	P	
3	0	0	0	3	20	20	-	60	-	100

SEE - Semester End Examination, **CIA** - Continuous Internal Assessment (It consists of Assignments/Seminars/Presentations/MCQ Tests, etc.)

Course Content

W - Weightage (%) , **T** - Teaching hours

Sr.	Topics	W	T
1	**Introduction to python programing:** Introduction to Python and basic programming concepts, variables, data types, conditionals statements and loops Lists,Sets,Tuples,Dictionaries: Working with strings, lists, sets, tuples and dictionaries, including common operations and built-in functions	15	6
2	**Functions :** Defining and using functions, including the use of arguments and return values OOPS Concepts : Object, class, abstraction, encapsulation, polymorphism, Inheritence. Exceptions and File handling: Handling exceptions and working with files	20	5
3	**Modules and Packages:** Working with modules and packages in Python Introduction to popular Python libraries for specific tasks, such as data analysis, web development, or game development. PyCharm IDE : GIT- Git Integration with PyCharm IDE, PyTests. Python connectivity with Databases MYSQL, MongoDB CRUD operations.	15	5
4	**Flask Framework:** Introduction to Flask and web development with Python, Installation in Virtual Environment. Creation Routing App Settings URL Building HTTP methods Templates Working with Static, Media Files. Sending Form Data to Template. Flask App with Database connectivity Sqlite3, MySQL. Handling Exceptions and Errors Flash Message Working with Mails. Authenticating and authorizing users with Flask-Login, Deploying a Flask application to a web server.	20	10
5	**Django Framework:** Introduction to Django framework, Django Project Installation in Virtual Environment. Phases in Django Project Creation Create a Project. Creation of Apps and their Structure. Working with ADMIN Console. Creating Views URL Mapping. Template System Working with Models. Form Processing static, media files, Django App Deployment.	20	10
6	**RESTful APIs:** Introduction to RESTful APIs and the REST architectural style Understanding the HTTP protocol and its role in RESTful APIs Designing and implementing RESTful APIs using common HTTP methods, such as GET, POST, PUT, and DELETE Using URLs and resource representations to identify and transfer data in RESTful APIs Implementing best practices for designing and implementing RESTful APIs, such as using HTTP status codes,	10	6

	versioning, and error handling Consuming RESTful APIs using common tools and libraries, such as cURL, Postman, and the requests library in Python Building scalable and secure RESTful APIs using common frameworks and libraries Flask or FastAPI.		

Reference Books

1.	**Fluent Python, 2nd Edition by Luciano Ramalho (TextBook)**
2.	**Learn Python3 the Hard Way By Zed Shaw**
3.	**"Django for Beginners: Build websites with Python and Django" by William S. Vincent.**
4.	**"Learning Django Web Development" by Samuli Natri.**
5.	**"Flask Web Development with Python" by Miguel Grinberg.**
6.	**"Mastering Flask" by Jack Stouffer.**
7.	**"Building RESTful Python Web Services" by Gastón C. Hillar.**
8.	**Building Web APIs with FastAPI" by Samuel Colvin.**

Course Outcome

After Learning the Course the students shall be able to:
After learning this course students are able to: 1. Understand the fundamental concepts of web development. 2. Create and manipulate data using a variety of databases, including SQL and NoSQL 3. Build and deploy web applications using a popular Python web framework, such as Django or Flask. 4. Design and implement APIs (application programming interfaces) that enable different applications to communicate with each other. 5. Test and debug web applications, and to deploy them to production environments.