Full Stack Development Syllabus

With Python

Introduction to Full Stack Development

- ✓ Overview of full-stack development
- ✓ Role of a full stack developer
- ✓ Technologies used in Full Stack Development (front-end, back-end, databases)
- ✓ Setting up the development environment

Front-End Development

- ✓ **HTML5:**
  - Structure of a webpage, semantic elements
  - Forms, tables, and input elements
  - HTML5 new features (video, audio, etc.)
- ✓ **CSS3:**
  - Styling basics: text, layout, colors, and typography
  - CSS Flexbox and Grid for responsive design
  - CSS animations and transitions
  - Preprocessors like Sass or Less
- ✓ **JavaScript (ES6+):**
  - Basic concepts: variables, data types, operators
  - Functions, objects, arrays, loops
  - DOM manipulation and event handling
  - Introduction to ES6 features: arrow functions, promises, async/await, destructuring, etc.

- ✓ **Responsive Web Design**:
    - Media queries and mobile-first approach
    - Bootstrap or Materialize framework
    - Progressive enhancement and adaptive design
- ✓ **Frontend Frameworks/Libraries**:
    - **React.js**:
        - JSX syntax, components, and props
        - State management with hooks
        - Component lifecycle
        - React Router for navigation
        - Introduction to state management (e.g., Redux, Context API)

Back-End Development with Python

- ✓ **Introduction to Python**:
    - Variables, data types, loops, and control structures
    - Functions, classes, and modules
    - File handling and exception handling
    - Working with libraries (requests, json, etc.)
- ✓ **Web Development Frameworks**:
    - **Flask** (lightweight framework):
        - Setting up a Flask application
        - Routing, templates (Jinja2)
        - Handling HTTP requests and responses
        - RESTful APIs with Flask
        - Middleware and error handling

- **Django** (full-stack framework):
  - Setting up Django projects and apps
  - Models, Views, Templates (MVT architecture)
  - Django ORM (Object-Relational Mapping)
  - Django Admin interface
  - User authentication and authorization
  - URL routing and form handling
  - Django Rest Framework (DRF) for creating APIs

Database Integration

✓ **Relational Databases** (SQL):

- Introduction to SQL and relational databases
- Creating and managing tables, primary and foreign keys
- Basic queries: SELECT, INSERT, UPDATE, DELETE
- Joins, aggregations, and subqueries
- SQLAlchemy ORM (for Flask) and Django ORM

✓ **NoSQL Databases:**

- Introduction to NoSQL databases (e.g., MongoDB)
- CRUD operations in MongoDB
- Mongoose for Node.js or PyMongo for Python

✓ **Database Relationships**:

- One-to-one, one-to-many, and many-to-many relationships
- Schema design and normalization

APIs and Web Services

✓ **RESTful APIs:**

- REST principles (stateless, client-server, uniform interface)
- HTTP methods (GET, POST, PUT, DELETE, PATCH)
- JSON and XML for data exchange
- Flask/Django for API creation

✓ **GraphQL:**

- Introduction to GraphQL and queries
- Setting up a basic GraphQL API using Python libraries (e.g., Graphene)

✓ **Authentication & Authorization:**

- Session-based authentication
- Token-based authentication (JWT)
- OAuth 2.0 for third-party login (Google, Facebook, etc.)
- User roles and permissions

Version Control and Collaboration

✓ **Git:**

- Git basics (cloning, committing, pushing, pulling)
- Branching, merging, and resolving conflicts
- GitHub for repository management
- Collaboration using pull requests

✓ **Code Collaboration Platforms:**

- GitHub/GitLab/Bitbucket for version control and code review
- Project management tools (e.g., Jira, Trello)

Testing and Debugging

✓ **Unit Testing:**
- Introduction to testing frameworks (e.g., PyTest, Unittest)
- Writing test cases for Python applications
- Mocking and patching external dependencies

✓ **End-to-End Testing:**
- Tools like Selenium or Cypress for UI testing
- Writing integration tests for APIs and databases

✓ **Debugging:**
- Debugging techniques using Python debugger (pdb)
- Logging errors and exceptions

Deployment and Cloud Integration

✓ **Deployment:**
- Hosting on platforms like Heroku, AWS, or DigitalOcean
- Configuring a production environment (e.g., WSGI, Gunicorn, Nginx)
- Continuous integration/continuous deployment (CI/CD)
- Docker for containerizing Python applications

✓ **Cloud Services:**
- Using AWS, GCP, or Azure for cloud computing
- S3 for file storage, RDS for managed databases
- Serverless architecture with AWS Lambda

- ✓ **Containerization with Docker**:
  - Docker basics (images, containers)
  - Creating Dockerfiles for Python applications
  - Docker Compose for multi-container applications

Web Sockets and Real-Time Applications

- ✓ Introduction to WebSockets for real-time communication
- ✓ Using **Socket.IO** with Flask or Django
- ✓ Building a chat application or a live notifications system

Advanced Topics (Optional)

- ✓ **Microservices Architecture**:
  - Building microservices with Flask/Django
  - Interservice communication via REST/GraphQL
- ✓ **Asynchronous Programming**:
  - Introduction to async/await in Python
  - Async frameworks like FastAPI for building high-performance APIs
- ✓ **Serverless Architecture**:
  - Using AWS Lambda or Google Cloud Functions
  - Building serverless APIs with Flask or FastAPI

Project Development

- ✓ Building a full-stack web application with Python (e.g., a blog or e-commerce site)
- ✓ Combining Flask/Django back end with React front end
- ✓ Incorporating databases and authentication
- ✓ Deploying the application to the cloud and ensuring scalability