Backend Development with Python

Comprehensive Training Course Outline

Course Overview

Duration: 4 Days (28 hours)
Format: Interactive lectures, hands-on exercises, and live implementation
Delivery Mode: Online (Remote Training)
Level: Intermediate
Prerequisites: Python programming experience

Module 1.1: Introduction to Backend Development

Learning Objectives:

- Understand the role of backend development in modern web applications
- Identify key responsibilities of backend developers
- Explore the Python ecosystem for backend development

Topics Covered:

- What is backend development?
- Server-side vs. client-side programming
- The role of Python in backend development
- Overview of popular Python backend frameworks
- Current industry trends and best practices

Hands-on Activity:

- Group discussion: Analyzing real-world web applications and their backend requirements

---

Module 1.2: Understanding How Websites and Web Applications Work

Learning Objectives:

- Comprehend the request-response cycle
- Understand HTTP/HTTPS protocols
- Learn about web servers and application servers

Topics Covered:

- Client-server architecture
- HTTP methods (GET, POST, PUT, DELETE, PATCH)
- Status codes and headers
- Cookies and sessions
- RESTful architecture principles
- WebSockets and real-time communication

Hands-on Activity:

- Using browser developer tools to analyze HTTP requests
- Examining API calls from popular websites

Module 1.3: Installing Python Packages and Libraries

Learning Objectives:

- Master Python package management
- Set up virtual environments
- Install and manage dependencies

Topics Covered:

- Python package managers (pip, conda)
- Creating and managing virtual environments (venv, virtualenv)
- Requirements.txt and dependency management
- Installing Flask, Django, and common libraries
- Understanding package versioning

Hands-on Activity:

- Creating a virtual environment
- Installing Flask and essential packages
- Setting up a requirements.txt file

Module 1.4: Preparing Your Backend Development Environment

Learning Objectives:

- Configure an optimal development environment
- Set up essential development tools
- Establish coding standards and best practices

Topics Covered:

- IDE selection and configuration (VS Code, PyCharm)
- Debugger setup and usage
- Linting and code formatting (pylint, black, flake8)
- Environment variables and configuration management
- Development vs. production environments
- Docker basics for development consistency

Hands-on Activity:

- Setting up a complete development environment
- Configuring VS Code with Python extensions
- Creating a project structure template

Module 1.5: Understanding Presentation Layer vs. Server Side

Learning Objectives:

- Differentiate between frontend and backend responsibilities
- Understand separation of concerns
- Learn about API-driven architecture

Topics Covered:

- Frontend vs. backend responsibilities
- Model-View-Controller (MVC) pattern
- API-first development approach
- Single Page Applications (SPAs) and backend APIs
- Microservices architecture overview

Module 1.6: Python Fundamentals Review

Learning Objectives:

- Refresh essential Python concepts for backend development
- Master Python features commonly used in web development

Topics Covered:

- Object-oriented programming in Python
- Decorators and their use in web frameworks
- Context managers and file handling
- Exception handling and error management
- Asynchronous programming basics (async/await)
- List comprehensions and generators
- Working with JSON and XML data

Hands-on Activity:

- Building utility functions for web applications
- Creating custom decorators for route handling

Module 2.1: Databases and SQL Fundamentals

Learning Objectives:

- Understand relational database concepts
- Write efficient SQL queries
- Learn database design principles

Topics Covered:

- Relational database concepts
- SQL basics: SELECT, INSERT, UPDATE, DELETE
- Joins, aggregations, and subqueries
- Database normalization and design patterns
- Indexes and query optimization
- Transactions and ACID properties
- ORMs vs. raw SQL

Hands-on Activity:

- Designing a database schema for a web application
- Writing SQL queries for common operations

- Using SQLite for development

Module 2.2: Linux Fundamentals

Learning Objectives:

- Navigate Linux command line effectively
- Understand Linux file system and permissions
- Perform basic server administration tasks

Topics Covered:

- Essential Linux commands for developers
- File permissions and ownership
- Process management
- Package management (apt, yum)
- SSH and remote server access
- Log file management
- Cron jobs for scheduled tasks

Hands-on Activity:

- Navigating and managing files via command line
- Setting up SSH keys
- Monitoring system resources

Module 2.3: Choosing a Development Framework

Learning Objectives:

- Compare popular Python web frameworks
- Select appropriate framework for project requirements

Topics Covered:

- Flask vs. Django vs. FastAPI
- Micro-frameworks vs. full-stack frameworks
- Framework ecosystem and community support
- Performance considerations
- Learning curve and development speed

Hands-on Activity:

- Framework comparison exercise
- Decision matrix for framework selection

Module 2.4: Introduction to Flask

Learning Objectives:

- Understand Flask architecture and philosophy
- Create basic Flask applications
- Master routing and view functions

Topics Covered:

- Flask application structure
- Routing and URL patterns
- Request and response objects
- Flask configuration management
- Application factory pattern
- Blueprints for modular applications

Hands-on Activity:

- Creating your first Flask application
- Building multiple routes with different HTTP methods
- Implementing basic error handling

Module 2.5: Setting up a Web Application Server

Learning Objectives:

- Configure web servers for Python applications
- Understand WSGI and application servers

Topics Covered:

- Apache vs. Nginx for Python applications

- WSGI servers (Gunicorn, uWSGI)
- Reverse proxy configuration
- SSL/TLS certificate setup
- Load balancing basics
- Server hardening and security
- Linux + Nginx + Python stack configuration

Hands-on Activity:

- Installing and configuring Nginx
- Deploying Flask application with Gunicorn
- Setting up reverse proxy

Module 2.6: Handling User Input

Learning Objectives:

- Process form data securely
- Validate and sanitize user input
- Handle file uploads

Topics Covered:

- Processing GET and POST requests
- Form handling with Flask-WTF
- Input validation and sanitization
- CSRF protection
- File upload handling and validation
- JSON request handling
- Query parameters and path variables

Hands-on Activity:

- Creating forms with validation
- Building file upload functionality
- Implementing secure input processing

Module 3.1: Generating Output and Using Templates

Learning Objectives:

- Use Jinja2 templating engine effectively
- Create dynamic HTML responses
- Implement template inheritance

Topics Covered:

- Jinja2 template syntax
- Template inheritance and includes
- Passing data to templates
- Template filters and custom filters
- Macros for reusable components
- Static file handling (CSS, JavaScript, images)
- Content Security Policy (CSP)

Hands-on Activity:

- Creating a base template layout
- Building reusable template components
- Implementing dynamic page generation

Module 3.2: Connecting to a Database

Learning Objectives:

- Integrate databases with Flask applications
- Use SQLAlchemy ORM effectively
- Implement database migrations

Topics Covered:

- Flask-SQLAlchemy setup and configuration
- Defining database models
- Relationships (one-to-many, many-to-many)
- Database queries with SQLAlchemy
- Flask-Migrate for database migrations
- Connection pooling and performance
- Database transactions

Hands-on Activity:

- Creating database models for a blog application
- Implementing CRUD operations
- Running database migrations

Module 3.3: Building RESTful APIs

Learning Objectives:

- Design RESTful API endpoints
- Implement proper HTTP methods and status codes
- Create API documentation

Topics Covered:

- RESTful API design principles
- Resource-based URL design
- JSON serialization and deserialization
- Flask-RESTful extension
- API versioning strategies
- Rate limiting and throttling
- API authentication (JWT, OAuth2)
- CORS handling
- API documentation with Swagger/OpenAPI

Hands-on Activity:

- Building a complete RESTful API
- Implementing authentication and authorization
- Testing API endpoints with Postman

Module 3.4: Enabling Users to Register through the Application

Learning Objectives:

- Implement secure user authentication
- Manage user sessions
- Handle password security

Topics Covered:

- User registration workflow
- Password hashing with bcrypt
- Flask-Login for session management
- Remember me functionality
- Password reset functionality
- Email verification
- Two-factor authentication (2FA)
- Social authentication (OAuth)

Hands-on Activity:

- Building complete user authentication system
- Implementing password reset via email
- Creating protected routes

Module 3.5: Securing the Web Application

Learning Objectives:

- Identify common security vulnerabilities
- Implement security best practices
- Protect against common attacks

Topics Covered:

- OWASP Top 10 vulnerabilities
- SQL injection prevention
- Cross-Site Scripting (XSS) prevention
- CSRF protection
- Secure headers configuration
- Input validation and sanitization
- Security testing tools

Hands-on Activity:

- Security audit of existing application
- Implementing security best practices
- Testing for common vulnerabilities

Module 3.6: Working with NoSQL Databases

Learning Objectives:

- Understand NoSQL database concepts
- Integrate MongoDB with Flask
- Choose between SQL and NoSQL

Topics Covered:

- NoSQL database types and use cases
- MongoDB basics
- Flask-PyMongo integration
- Document design patterns
- When to use NoSQL vs. SQL
- Redis for caching and sessions
- Elasticsearch for search functionality

Hands-on Activity:

- Connecting Flask to MongoDB
- Implementing CRUD operations with MongoDB
- Adding Redis caching layer

Module 4.1: Testing the Web Application

Learning Objectives:

- Write unit tests for Flask applications
- Implement integration tests
- Set up continuous testing

Topics Covered:

- Unit testing with pytest
- Flask test client
- Mocking and fixtures
- Test coverage analysis
- Integration testing
- End-to-end testing basics
- Test-driven development (TDD)

Hands-on Activity:

- Writing unit tests for routes and models
- Setting up pytest configuration
- Achieving high test coverage

Module 4.2: Managing the Project Using Version Control

Learning Objectives:

- Master Git for backend projects
- Implement branching strategies
- Collaborate effectively with team members

Topics Covered:

- Git fundamentals review
- Branching strategies (Git Flow, GitHub Flow)
- Pull requests and code reviews
- Resolving merge conflicts
- .gitignore for Python projects
- Git hooks for automation
- GitHub/GitLab CI/CD integration

Hands-on Activity:

- Setting up Git repository
- Implementing feature branch workflow
- Conducting code review exercise

Module 4.3: Deployment Techniques and Continuous Integration

Learning Objectives:

- Set up continuous integration and deployment
- Automate testing and deployment processes
- Deploy to cloud platforms

Topics Covered:

- CI/CD concepts and benefits
- GitHub Actions / GitLab CI setup
- Automated testing in CI pipeline
- Deployment strategies (blue-green, canary)
- Docker containerization
- Deployment to cloud platforms (AWS, Heroku, DigitalOcean)

- Environment-specific configurations
- Rollback strategies

Hands-on Activity:

- Creating CI/CD pipeline with GitHub Actions
- Containerizing Flask application with Docker
- Deploying to a cloud platform

Module 4.4: Monitoring Application Performance

Learning Objectives:

- Implement application monitoring
- Set up logging and error tracking
- Use performance monitoring tools

Topics Covered:

- Application logging best practices
- Centralized logging (ELK stack, CloudWatch)
- Error tracking with Sentry
- Application performance monitoring (APM)
- Metrics and alerting
- Health check endpoints
- Monitoring dashboards

Hands-on Activity:

- Implementing structured logging
- Setting up error tracking
- Creating monitoring dashboard

Module 4.5: Optimizing the Web Application

Learning Objectives:

- Identify performance bottlenecks
- Implement optimization techniques
- Use caching effectively

Topics Covered:

- Profiling Flask applications
- Database query optimization
- Caching strategies (Redis, Memcached)
- Static file optimization
- Compression and minification
- Asynchronous task processing (Celery)
- CDN integration

Hands-on Activity:

- Profiling application performance
- Implementing caching layer
- Optimizing database queries

Module 4.6: Scaling a Web Application

Learning Objectives:

- Understand scaling strategies
- Design for horizontal scalability
- Implement load balancing

Topics Covered:

- Vertical vs. horizontal scaling
- Stateless application design
- Load balancing strategies
- Database scaling techniques
- Microservices architecture
- Message queues for async processing
- Auto-scaling considerations

Module 4.7: Frontend Integration

Learning Objectives:

- Integrate backend with modern frontend frameworks
- Handle CORS and authentication
- Build full-stack applications

Topics Covered:

- Serving API to React/Angular/Vue.js
- CORS configuration
- Token-based authentication for SPAs
- WebSockets for real-time features
- Server-Sent Events (SSE)
- API design for frontend consumption
- Handling file uploads from frontend

Hands-on Activity:

- Creating API endpoints for frontend consumption
- Testing integration with a React frontend
- Implementing real-time features

Module 4.8: Expanding the Application using Advanced Python Features

Learning Objectives:

- Apply advanced Python concepts
- Write more efficient code
- Use Python features for better applications

Topics Covered:

- Decorators for route protection and logging
- Context managers for resource management
- Generators for streaming responses
- Async/await for concurrent operations
- Metaclasses and their uses
- Type hints for better code quality
- Python 3.10+ features

Hands-on Activity:

- Implementing custom decorators
- Building streaming endpoints
- Using type hints with mypy

Module 4.9: Troubleshooting

Learning Objectives:

- Debug common issues effectively
- Use debugging tools and techniques

Topics Covered:

- Common Flask errors and solutions
- Debugging techniques
- Reading stack traces
- Database debugging
- Performance debugging
- Production troubleshooting

Hands-on Activity:

- Debugging session with real-world scenarios
- Using debugging tools effectively

Module 4.10: Summary and Conclusion

Learning Objectives:

- Consolidate learning from the entire course
- Identify next steps for continued growth

Topics Covered:

- Course recap and key takeaways
- Best practices summary
- Resources for continued learning
- Next steps for participants
- Q&A session
- Course evaluation and feedback