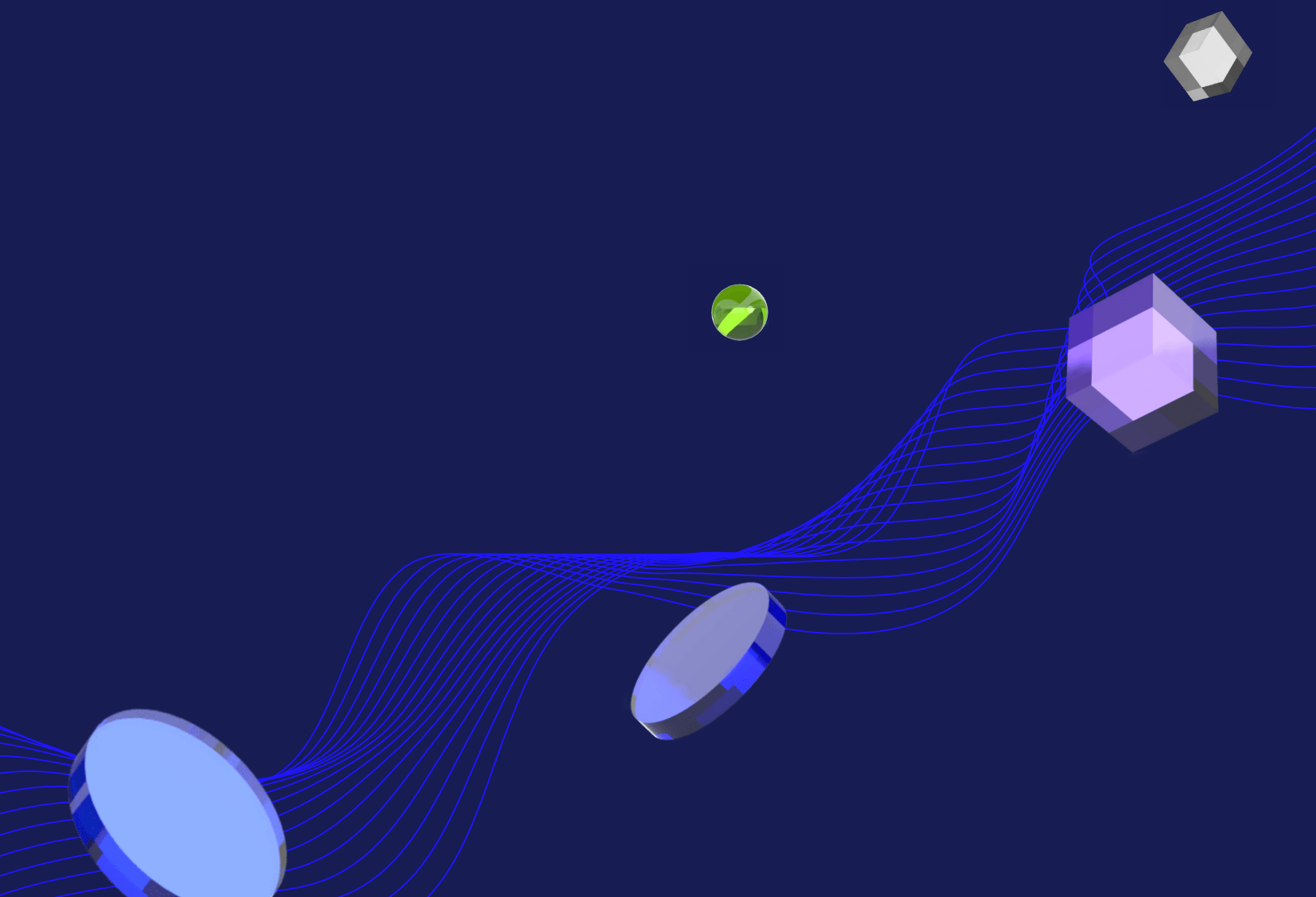
INDIVIDUAL LEARNERS

UDACITY

SCHOOL OF PROGRAMMING & DEVELOPMENT

Full Stack Web Developer

Nanodegree Program Syllabus



Overview

The goal of the Full Stack Web Developer Nanodegree program is to equip learners with the unique skills they need to build database-backed APIs and web applications. Each project will be an opportunity to apply the lessons learned and demonstrate skills to potential employers.

Learning Objectives

A graduate of this program will be able to:

- Design and build a database for a software application.
- Create and deploy a database-backed web API (Application Programming Interface).
- Secure and manage user authentication and access control for an application backend.
- Deploy a Flask-based web application to the cloud using Docker and Kubernetes.

Program information

Estimated Time

4 months at 5-10hrs/week*

Skill Level

Intermediate

Prerequisites

Learners must be able to:

- Write and test software with Python or another object-oriented programming language.
- Query a SQL database using SELECT.
- Write to a SQL database using INSERT.
- Write software for front end applications and websites using JavaScript to:
  - Fetch and display data from an API using AJAX or Fetch.
  - Organize data using JSON (JavaScript Object Notation).

Required Hardware/Software

Learners need a computer with a broadband internet connection. This program uses Python 3.7, PostgreSQL 11, SQLAlchemy, Flask 1.0, Docker and various Python packages.

*The length of this program is an estimation of total hours the average student may take to complete all required coursework, including lecture and project time. If you spend about 5-10 hours per week working through the program, you should finish within the time provided. Actual hours may vary.

Course 1

SQL & Data Modeling for the Web

Master relational databases with the power of SQL, and leverage Python to incorporate database logic into programs.

Course Project

Design a Venue Booking Database

For the first project, learners will be building out the data models and database for an artist/venue booking application. The fictitious startup Fy-yur is building a website that facilitates bookings between artists who can play at venues, and venues who want to book artists. This site:

- Lets venue managers and artists sign up, fill out their information, and list their availability for shows.
- Lets artists browse venues where they can play, and see what past/upcoming artists have been booked at a venue.
- Lets a venue manager browse artists that would like to play in their city, and see what past/upcoming venues where the artist has played/will be playing.

The goal of this project is to build out the data models for this booking application. A prototype design of the web app will be provided. Learners will use SQLAlchemy and Postgresql to build out the data models upon which this site will rely. They'll write out both the raw SQL and SQLAlchemy commands to run for powering the backend functionality of the website.

Lesson 1

Connecting & Interacting with Databases

- Describe and explain the client-server model.
- Describe and explain the TCP/IP communication protocol.
- Describe and explain the base unit of database work: transactions.
- Install the PostgreSQL database management system.
- Create and manage Postgres databases with the psql client.
- Install the psycopg2 Python+Postgres database driver.
- Create and manage Postgres databases using the psycopg2 Python database driver.

Lesson 2

Intro to SQLAlchemy & SQLAlchemy ORM Basics

- Describe and explain the use cases for an Object Relational Mapping (ORM) library.
- Describe and explain the abstraction layers of SQLAlchemy.
- Connect to and manage a database using composable SQL expressions.
- Define data model objects with Python using SQLAlchemy ORM.
- Connect data models to a lightweight Flask web application.
- Build data models using different types of data.

Lesson 3

SQLAlchemy ORM in Depth

- Explore and retrieve data using the SQLAlchemy Model query object.
- Create database sessions for executing database transactions.
- Execute database transactions within a connection session.
- Describe and explain the SQLAlchemy object lifecycle.
- Build a lightweight data app using SQLAlchemy.
- Describe and explain the Model-View-Controller (MVC) application architecture.
- Retrieve from data from a webform using Flask.
- Update data models using data migrations.
- Migrate data using Flask-Migrate and Flask-Script.
- Define and code relationships between tables and objects using SQLAlchemy.
- Implement database methods to query relationships between data models.

Lesson 4

Build A Crud App with SQLAlchemy ORM: Part 1

- Use the CRUD (Create, Read, Update, Delete) model to build a small database backed app.
- Capture user input from a webform to add and modify data to a database.
- Manage data using database sessions in an application controller.

Lesson 5

Migrations

- Modify a data schema using Flask-Migrate and Alembic.
- Write migration scripts to update data schemas using Flask-Script.

Lesson 6

Build a Crud App with SQLAlchemy ORM: Part 2

- Update database models using webforms and application routing.
- Delete information from a database using SQLAlchemy.
- Model and control relationships between different types of data objects.
- Implement one-to-many and many-to-many relationships using SQLAlchemy.
- Execute complex database queries on related data models.

Course 2

API Development and Documentation

Learn how to use APIs to control and manage web applications, including best practices for API testing and documentation.

Course Project

Trivia API

In this project, learners will use the skills they've developed to build a Trivia API. The API will allow users to:

- Search for trivia questions and answers via category and difficulty.
- Add new questions.
- Modify the difficulty rating of questions.

The goal of this project is to use APIs to control and manage a web application using existing data models. Learners will be given a set of data models and the application front end. Their task will be to implement the API in Flask to make the Trivia game functional.

Lesson 1

Introduction to APIs

- Describe and explain the definition and use cases of APIs (application programming interface).
- Describe and explain how APIs are used to connect application front ends to server backends.

Lesson 2

HTTP & Flask Basics

- Describe and explain the Hypertext Transfer Protocol (HTTP).
- Describe and explain the components of an HTTP request.
- Describe and explain the different HTTP methods (verbs).
- Describe and explain HTTP status codes.
- Request information from a server using cURL and HTTP requests.
- Install the Python Flask micro application framework.
- Set up and configure a Flask application.
- Create a Flask endpoint (route).

Lesson 3

Endpoints and Payloads

- Structure and organize API Endpoints.
- Describe and explain Cross-Origin Resource Sharing (CORS).
- Manage CORS requests using HTTP headers.
- Manage CORS controls using Flask-CORS.
- Parse request path and body from an HTTP request.
- Implement HTTP POST, PATCH, and DELETE methods using Flask.
- Handle application errors using Flask.

Lesson 4

API Testing

- Describe and explain the purpose and benefits of API testing.
- Test a REST API using Flask and unittest.
- Develop an application iteratively and safely using Test Driven Development (TDD).

Lesson 5

API Documentation

- Read and explore API documentation from a number of API developers.
- Write effective documentation for your own API.

Course 3

Identity Access Management

Implement authentication and authorization in Flask and understand how to design against key security principle. Learners will also gain experience with role-based control design patterns, securing a REST API, and applying software system risk and compliance principles.

Course Project

Identity Access Management

In the third project of the program, learners will build the backend for a coffee shop application. They'll add user accounts and authentication to your application and use role-based access management strategies to control different types of user behavior in the app. The application must:

For this project, learners will use:

- Display graphics representing the ratio of ingredients in each drink.
- Allow public users to view drink names and graphics.
- Allow the shop baristas to see the recipe information.
- Allow the shop managers to create new drinks and edit existing drinks.

This project will give them a hands-on chance to practice and demonstrate their new skills, such as:

- Implementing authentication and authorization in Flask.
- Designing against key security principles.
- Implementing role-based control design patterns.
- Securing a REST API.
- Applying software system risk and compliance principles.

Lesson 1

Foundations

- Describe and explain the use cases and differences between authorization and authentication.
- Describe the problem of security and the risks of unsecured or improperly secured application systems.
- Describe different types of security attack.
- Inspect requests and responses for an application using Postman.

Lesson 2

Authentication

- Describe common methods for application authentication.
- Explain why passwords are not the ideal method for authentication.
- Implement an application authentication layer with Auth0.
- Secure API communications using JSON Web Tokens (JWT).

Lesson 3

Passwords

- Describe the risks associated with password controlled systems.
- Mitigate access risks associated with SQL injection by validating and sanitizing database inputs.
- Secure database data in a database using standard encryption practices.
- Describe how an attacker can use rainbow tables to gain access to a system.
- Improve security of hashed passwords and encrypted data using the ”salt” method.
- Increase application security by using best practices to avoid logging and serializing sensitive data.

Lesson 4

Authorization

- Describe the concept of authorization and access control.
- Define permissions in the context of an application.
- Constrain permissions in an application by using role-based access control (RBAC).
- Define permission roles using Auth0.
- Identify user permissions and roles from JWTs (JavaScript Web Tokens).

Lesson 5

Thinking Adversarially

- Prevent accidental access to privileged information in Git repositories by using environment variables.
- Mitigate risks to Git master branch changes by developing in feature branches.
- Employ code review as a practice to mitigate security risks.
- Test API and authentication practices with integration testing.
- Describe common types of adversarial attacks on network systems.

Course 4

Server Deployment & Containerization

Develop an understanding of containerized environments, use Docker to share and store containers, and deploy a Docker container to AWS Elastic Kubernetes Service using the CI/CD pipeline.

Course Project

Deploy a Flask App to Kubernetes Using EKS

In this project, learners will create a container for your Flask web app using Docker and deploy the container to a Kubernetes cluster using Amazon EKS. By the end of the project, they will have deployed their application live to the world, where it should be accessible by IP address. They'll use automated testing to prevent bad code from being deployed and monitor their app's performance using AWS tools.

Course Project

Capstone

In this final capstone project, learners will combine all of the new skills they've learned and developed in this course to construct a database-backed web API with user access control. Learners will choose what app to build and then they'll design and build out all of the API endpoints needed for the application and properly secure them for use in any front end application (web or mobile).

Lesson 1

Containers

- Describe and explain the benefits and use cases for containerized environments.
- Install Docker on a local machine.
- Define a container environment using a Dockerfile.
- Download and launch a Docker container.
- Store and share a docker container.

Lesson 2

AWS & Kubernetes

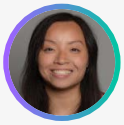
- Introduce AWS and commonly used services, S3, EC2, and IAM.
- Create AWS resources using both AWS console and AWS command-line interface (CLI).
- Describe and explain container orchestration, how it works, and the general use case.
- Describe and explain how Kubernetes manages container clusters.
- Creating an EKS Cluster using AWS console and Eksctl (CLI).
- Manage Kubernetes clusters using the Kubectl (CLI).

Lesson 3

Deployment Using CI/CD

- Deploy an application to the EKS cluster.
- Use Cloudformation to create AWS resources using a YAML script.
- Implement an end-to-end Continuous Integration (CI) and Continuous
- Delivery (CD) pipeline using AWS CodePipeline and AWS CodeBuild services.

Meet your instructors.



Amy Hua

Software Professional

Amy has 6+ years of experience as a software professional, building everything from data visualizations to self-driving cars. She's been a bootcamp instructor, StartupBus mentor, and Girls Who Code instructor.

Caryn McCarthy

Software Developer

Caryn has worked as a software developer and as coach and experience manager for Code Next at Google. She is passionate about diversity and equity in tech, is always working to create positive impact in the tech industry and the world.

Gabriel Ruttner

CTO at Ursa

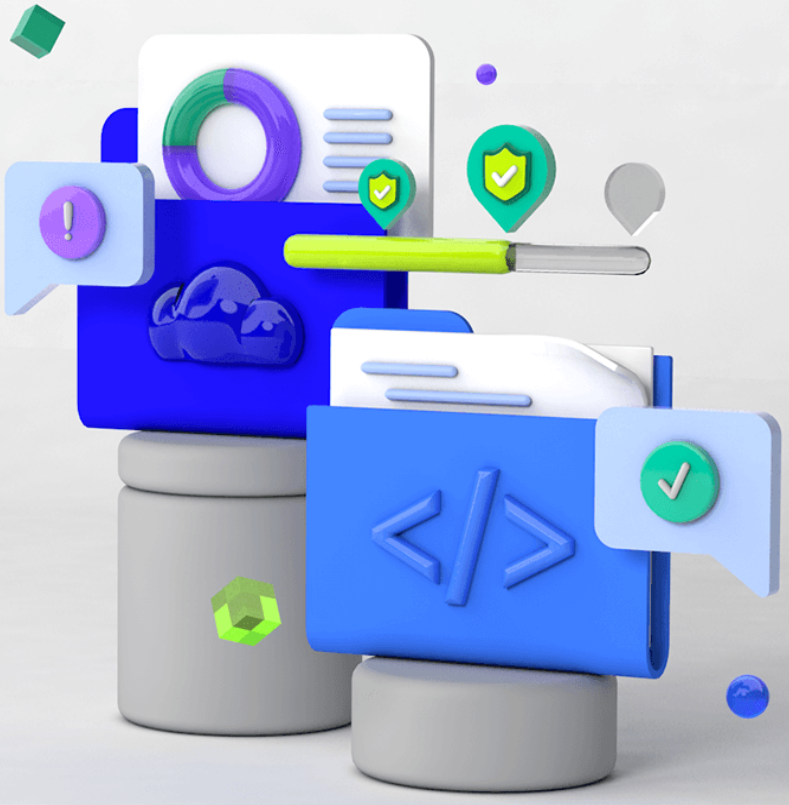
Gabe is the CTO at Ursa and a tech advisor for startups. He has expertise in building cloud-based machine learning and natural language processing services at early stage tech companies. He holds technical degrees from Cornell University and Stony Brook University.

Kennedy Behrman

Consultant & Author

Kennedy is a veteran consultant and author, specializing in architecting and implementing cloud solutions for early-stage startups. He is experienced in data engineering, data science, AWS solutions, and engineering management.

Udacity's learning experience



Hands-on Projects

Open-ended, experiential projects are designed to reflect actual workplace challenges. They aren't just multiple choice questions or step-by-step guides, but instead require critical thinking.

Quizzes

Auto-graded quizzes strengthen comprehension. Learners can return to lessons at any time during the course to refresh concepts.

Knowledge

Find answers to your questions with Knowledge, our proprietary wiki. Search questions asked by other students, connect with technical mentors, and discover how to solve the challenges that you encounter.

Custom Study Plans

Create a personalized study plan that fits your individual needs. Utilize this plan to keep track of movement toward your overall goal.

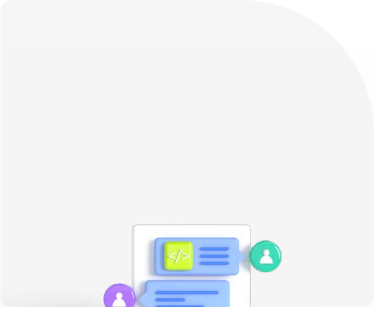
Workspaces

See your code in action. Check the output and quality of your code by running it on interactive workspaces that are integrated into the platform.

Progress Tracker

Take advantage of milestone reminders to stay on schedule and complete your program.

Our proven approach for building job-ready digital skills.



Experienced Project Reviewers

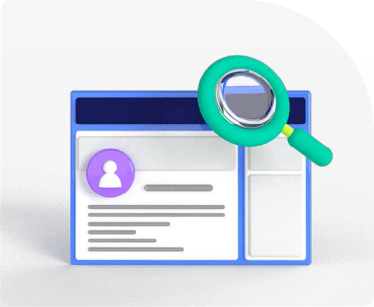
Verify skills mastery.

- Personalized project feedback and critique includes line-by-line code review from skilled practitioners with an average turnaround time of 1.1 hours.
- Project review cycle creates a feedback loop with multiple opportunities for improvement—until the concept is mastered.
- Project reviewers leverage industry best practices and provide pro tips.

Technical Mentor Support

24/7 support unblocks learning.

- Learning accelerates as skilled mentors identify areas of achievement and potential for growth.
- Unlimited access to mentors means help arrives when it's needed most.
- 2 hr or less average question response time assures that skills development stays on track.



Personal Career Services

Empower job-readiness.

- Access to a Github portfolio review that can give you an edge by highlighting your strengths, and demonstrating your value to employers.*
- Get help optimizing your LinkedIn and establishing your personal brand so your profile ranks higher in searches by recruiters and hiring managers.

Mentor Network

Highly vetted for effectiveness.

- Mentors must complete a 5-step hiring process to join Udacity's selective network.
- After passing an objective and situational assessment, mentors must demonstrate communication and behavioral fit for a mentorship role.
- Mentors work across more than 30 different industries and often complete a Nanodegree program themselves.

*Applies to select Nanodegree programs only.

UDACITY

Learn more at

www.udacity.com/online-learning-for-individuals →

01.13.23 | V1.0