UDACITY
FOR ENTERPRISE

THE SCHOOL OF PROGRAMMING AND DEVELOPMENT

Full Stack Web Developer

NANODEGREE SYLLABUS

Overview

Full Stack Web Developer Nanodegree Program

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. 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.

This program includes 4 courses and 5 projects. Each project you build will be an opportunity to apply what you've learned in the lessons and demonstrate that you have gained practical full-stack development skills.

Program Information

ESTIMATED TIME
4 months
Study 5-10 hours/week

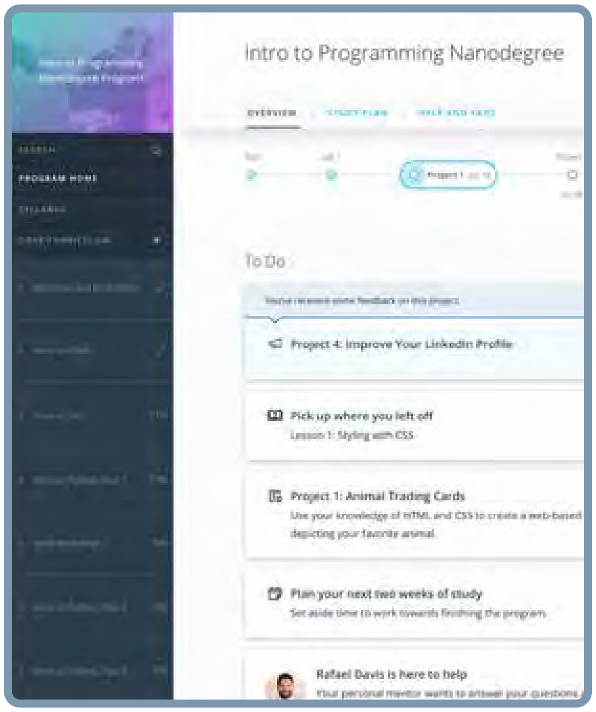
LEVEL
Practitioner

PREREQUISITES
• 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).

HARDWARE/SOFTWARE REQUIRED
A computer with a broadband internet connection. This program uses Python 3.7, PostgreSQL 11, SQLAlchemy, Flask 1.0, Docker and various Python packages.

LEARN MORE ABOUT THIS NANODEGREE
Contact us at enterpriseNDs@udacity.com.

Our Classroom Experience



You've received some feedback on this project

Project 4: Improve Your LinkedIn Profile

Pick up where you left off
Lesson 1: Styling with CSS

Project 1: Animal Trading Cards
Use your knowledge of HTML and CSS to create a web-based depicting your favorite animal.

Plan your next two weeks of study
Set aside time to work towards finishing the program.

Rafael Davis is here to help
Your personal mentor wants to answer your questions

REAL-WORLD PROJECTS

Learners build new skills through industry-relevant projects and receive personalized feedback from our network of 900+ project reviewers. Our simple user interface makes it easy to submit projects as often as needed and receive unlimited feedback.

KNOWLEDGE

Answers to most questions can be found with Knowledge, our proprietary wiki. Learners can search questions asked by others and discover in real-time how to solve challenges.

LEARNER HUB

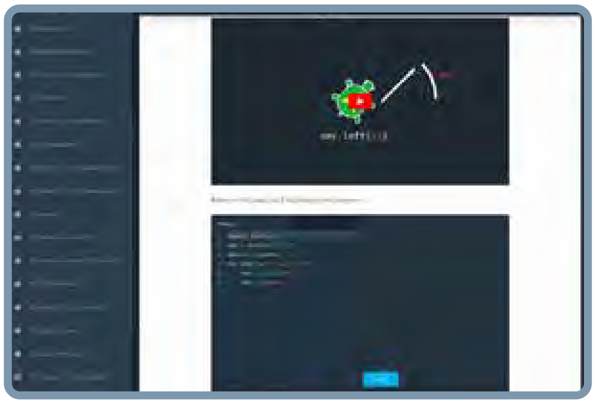
Learners leverage the power of community through a simple, yet powerful chat interface built within the classroom. Learner Hub connects learners with their technical mentor and fellow learners.

WORKSPACES

Learners can check the output and quality of their code by testing it on interactive workspaces that are integrated into the classroom.

QUIZZES

Understanding concepts learned during lessons is made simple with auto-graded quizzes. Learners can easily go back and brush up on concepts at anytime during the course.



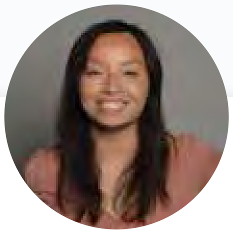
CUSTOM STUDY PLANS

Mentors create a custom study plan tailored to learners' needs. This plan keeps track of progress toward learner goals.

PROGRESS TRACKER

Personalized milestone reminders help learners stay on track and focused as they work to complete their Nanodegree program.

Learn with the Best



Amy Hua

INSTRUCTOR

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

INSTRUCTOR

Caryn has worked as a software developer and as Coach and Experience Manager at 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

INSTRUCTOR

Gabe is the CTO at Ursa & Tech Advisor for Start-Ups. 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

INSTRUCTOR

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.

Course 1: SQL and Data Modeling for the Web

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

Project Design a Venue Booking Database

For your first project, you'll 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. You'll use SQLAlchemy and Postgresql to build out the data models upon which this site will rely. You'll write out both the raw SQL and SQLAlchemy commands to run for powering the backend functionality of the website.

LESSON TITLE	LEARNING OUTCOMES
CONNECTING AND 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.

Course 1: SQL and Data Modeling for the Web, cont.

LESSON TITLE	LEARNING OUTCOMES
INTRO TO SQLALCHEMY AND 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.
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.

Course 1: SQL and Data Modeling for the Web, cont.

LESSON TITLE	LEARNING OUTCOMES
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.
MIGRATIONS	• Modify a data schema using Flask-Migrate and Alembic. • Write migration scripts to update data schemas using Flask-Script.
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.

```
BETWEEN 50001
    if ($anno != "") {
    $result1 = mysql_query($sql1);
    $result2 = mysql_query($sql2);
    $result3 = mysql_query($sql3);
    $result4 = mysql_query($sql4);
    $result5 = mysql_query($sql5);
    $result6 = mysql_query($sql6);
    $result11 = mysql_query($sql11);
    $result22 = mysql_query($sql22);
    $result33 = mysql_query($sql33);
    $result44 = mysql_query($sql44);
    $result55 = mysql_query($sql55);
    $result66 = mysql_query($sql66);
```

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.

Project Trivia API

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

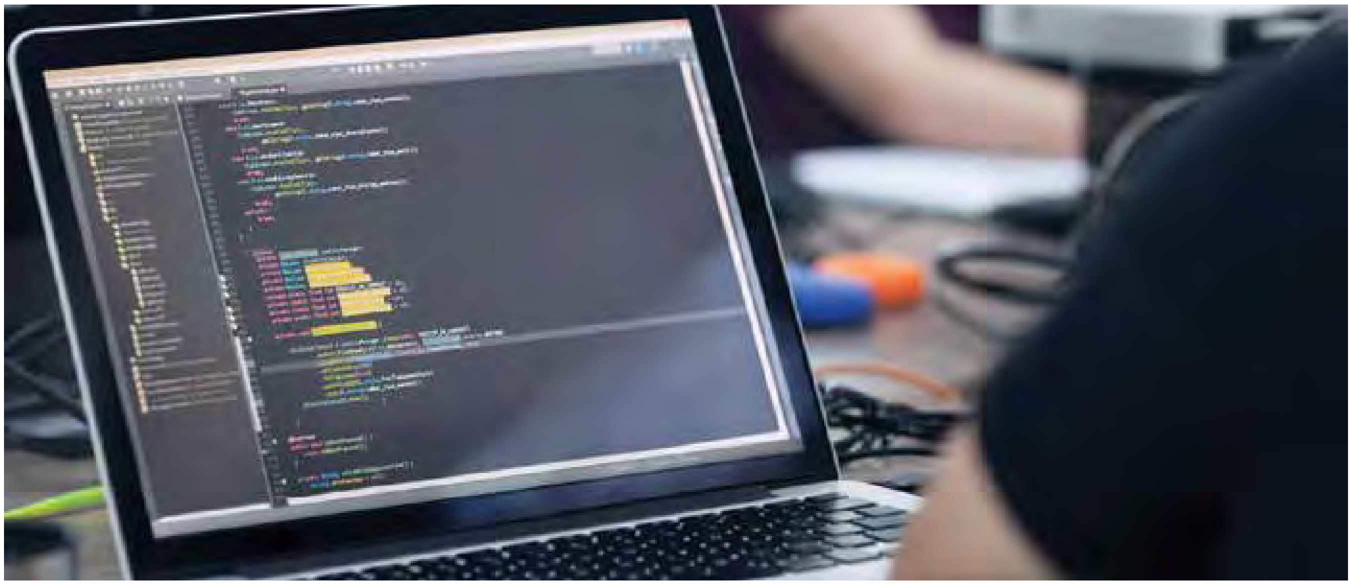
- Search for trivia questions and answers via category and difficulty.
- Add new questions.
- Modify the difficulty rating of questions.

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

LESSON TITLE	LEARNING OUTCOMES
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.
HTTP AND 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).

Course 2: API Development and Documentation, cont.

LESSON TITLE	LEARNING OUTCOMES
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.
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).
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. You will also gain experience with role-based control design patterns, securing a REST API, and applying software system risk and compliance principles.

Project Identity Access Management

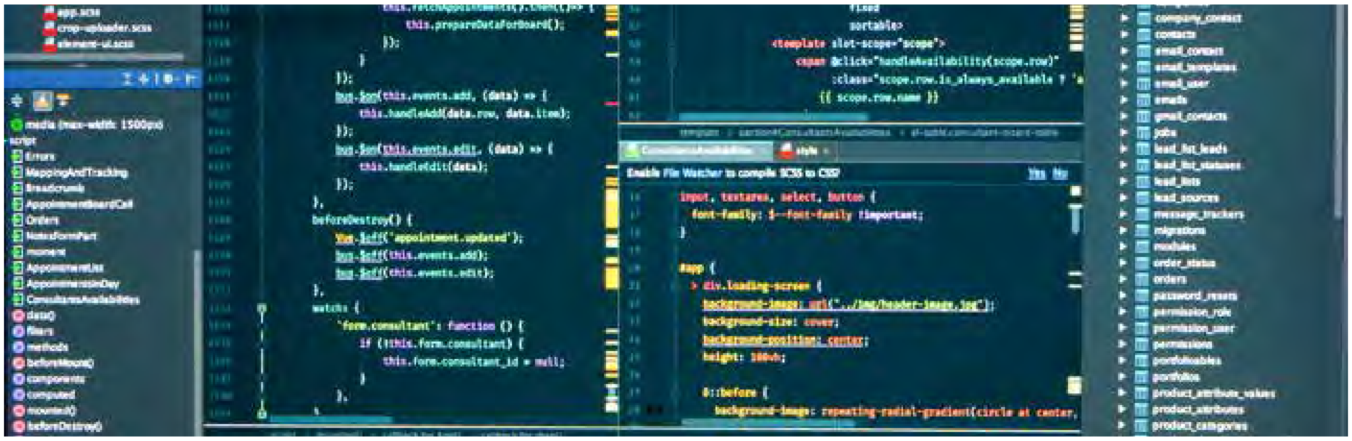
In the third project of the program, you will build the backend for a coffee shop application. You'll add user accounts and authentication to your application and use role-based access management strategies to control different types of user behavior in the app. The application must:

For this project, you 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 you a hands-on chance to practice and demonstrate your 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.



```
            this.prepareDataForBoard();
        });
    }
});
bus.$on(this.events.add, (data) => {
    this.handleAdd(data.row, data.item);
});
bus.$on(this.events.edit, (data) => {
    this.handleEdit(data);
});
},
beforeDestroy() {
    Vue.$off('appointment.updated');
    bus.$off(this.events.add);
    bus.$off(this.events.edit);
},
watch: {
    'form.consultant': function () {
        if (!this.form.consultant) {
            this.form.consultant_id = null;
        }
    },
```

```
                sortable>
            <template slot-scope="scope">
                <span @click="handleAvailability(scope.row)"
                      :class="scope.row.is_always_available ? 'a
                    {{ scope.row.name }}
```

Enable File Watcher to compile SCSS to CSS? Yes No

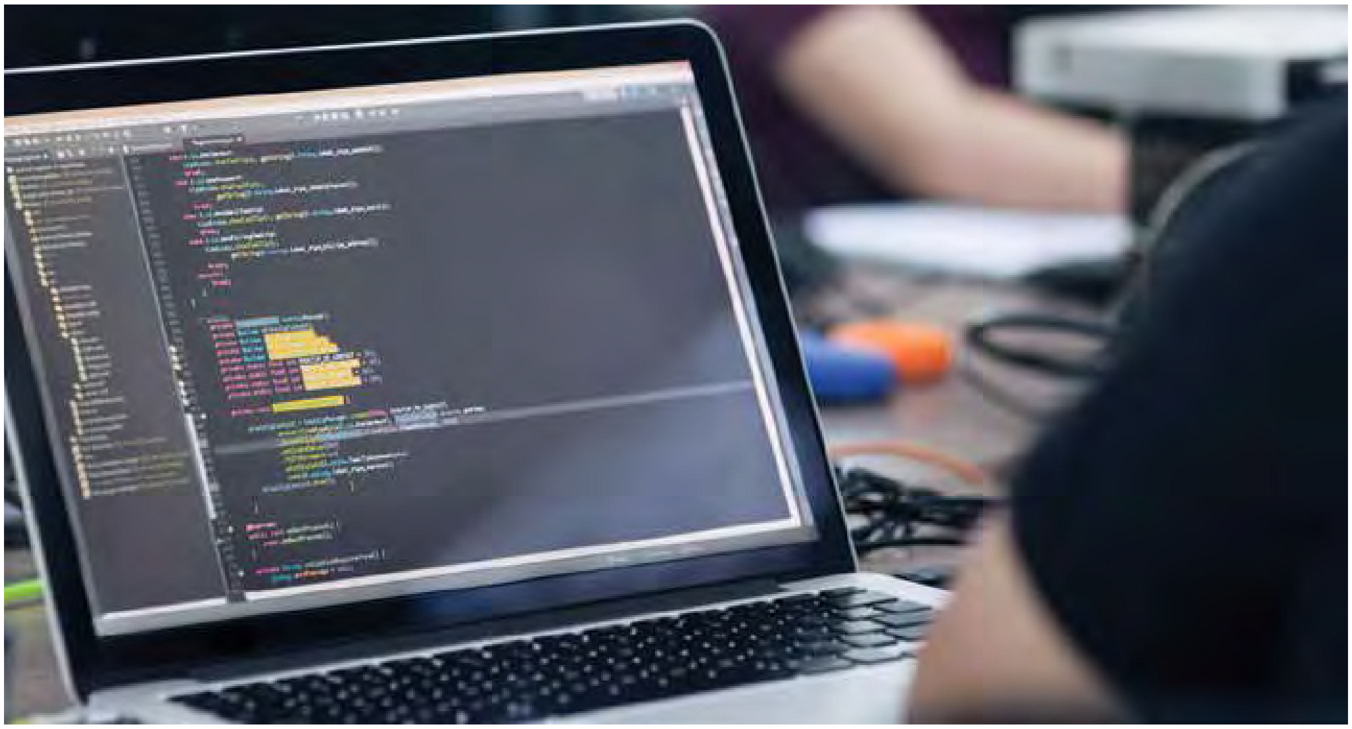
```
input, textarea, select, button {
  font-family: $--font-family !important;
}

#app {
  > div.loading-screen {
    background-image: url("../img/header-image.jpg");
    background-size: cover;
    background-position: center;
    height: 100vh;

    &::before {
      background-image: repeating-radial-gradient(circle at center,
```

Course 3: Identity Access Management, cont.

LESSON TITLE	LEARNING OUTCOMES
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.
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).
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.

Course 3: Identity Access Management, cont.

LESSON TITLE	LEARNING OUTCOMES
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).
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 and Containerization

Develop an understanding of containerized environments, use Docker to share and store containers, and deploy a Docker container to a Kubernetes cluster using AWS.

Project Deploy a Flask App to Kubernetes Using EKS

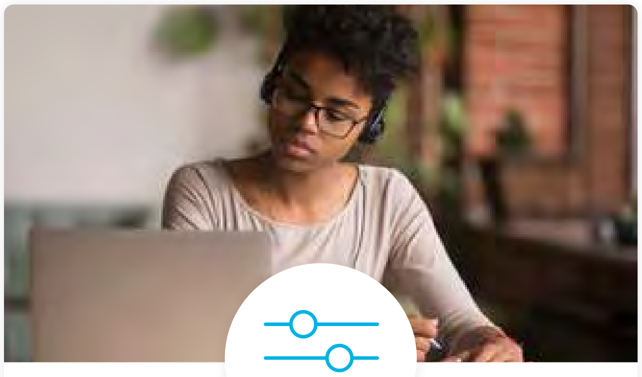
In this project, you will create a container for your Flask web app using Docker and deploy the container to a Kubernetes cluster using Amazon EKS. By the end of the project, you will have deployed your application live to the world, where it should be accessible by IP address. You'll use automated testing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

Project Capstone

In this final capstone project, you will combine all of the new skills you've learned and developed in this course to construct a database-backed web API with user access control. You will choose what app to build and then you'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 TITLE	LEARNING OUTCOMES
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.
DEPLOYMENT	• Describe and explain container orchestration, how it works and the general use case. • Describe and explain how Kubernetes manages container clusters. • Deploy a Docker container to a Kubernetes cluster using AWS EKS and the AWS command line interface (CLI). • Manage Kubernetes clusters using the AWS CLI. • Implement Continuous Delivery (CD) and Continuous Integration (CI) with AWS CodePipeline and AWS CodeBuild.

Our Nanodegree Programs Include:



Pre-Assessments

Our in-depth workforce assessments identify your team's current level of knowledge in key areas. Results are used to generate custom learning paths designed to equip your workforce with the most applicable skill sets.

Dashboard & Progress Reports

Our interactive dashboard (enterprise management console) allows administrators to manage employee onboarding, track course progress, perform bulk enrollments and more.

Industry Validation & Reviews

Learners' progress and subject knowledge is tested and validated by industry experts and leaders from our advisory board. These in-depth reviews ensure your teams have achieved competency.

Real World Hands-on Projects

Through a series of rigorous, real-world projects, your employees learn and apply new techniques, analyze results, and produce actionable insights. Project portfolios demonstrate learners' growing proficiency and subject mastery.

Our Review Process

Real-life Reviewers for Real-life Projects

Real-world projects are at the core of our Nanodegree programs because hands-on learning is the best way to master a new skill. Receiving relevant feedback from an industry expert is a critical part of that learning process, and infinitely more useful than that from peers or automated grading systems. Udacity has a network of over 900 experienced project reviewers who provide personalized and timely feedback to help all learners succeed.

Vaibhav
UDACITY LEARNER

"I never felt overwhelmed while pursuing the Nanodegree program due to the valuable support of the reviewers, and now I am more confident in converting my ideas to reality."

now at

CODING VISIONS INFOTECH

All Learners Benefit From:

- Line-by-line feedback for coding projects
- Industry tips and best practices
- Advice on additional resources to research
- Unlimited submissions and feedback loops

How it Works

Real-world projects are integrated within the classroom experience, making for a seamless review process flow.

- Go through the lessons and work on the projects that follow
- Get help from your technical mentor, if needed
- Submit your project work
- Receive personalized feedback from the reviewer
- If the submission is not satisfactory, resubmit your project
- Continue submitting and receiving feedback from the reviewer until you successfully complete your project

About our Project Reviewers

Our expert project reviewers are evaluated against the highest standards and graded based on learners' progress. Here's how they measure up to ensure your success.

900+
Expert Project Reviewers
Are hand-picked to provide detailed feedback on your project submissions.

1.8M
Projects Reviewed
Our reviewers have extensive experience in guiding learners through their course projects.

3
Hours Average Turnaround
You can resubmit your project on the same day for additional feedback.

4.85 /5
Average Reviewer Rating
Our learners love the quality of the feedback they receive from our experienced reviewers.

UDACITY
FOR ENTERPRISE