Full Stack Web Developer Nanodegree Syllabus

Build Complex Web Applications

UDACITY
FOR ENTERPRISE

Full Stack Web Developer

Table of Contents

Full Stack Web Developer

Before You Start

Thank you for your interest in the Full Stack Web Developer Nanodegree!
In order to succeed in this program, we recommend having experience programing in HTML, CSS and Python. If you've never written code before, we recommend starting with the Introduction to Programming Nanodegree Program, which will prepare you for this and other career-focused Nanodegree programs.

Contact Info

While going through the program, if you have questions about anything, you can reach us at enterprise-support@udacity.com. For help from Udacity Mentors and your peers, visit the Udacity Classroom.

Nanodegree Program Info

TECHNICAL REQUIREMENTS:

REQUIRED HARDWARE: Webcam, Microphone

REQUIRED SOFTWARE AND SOFTWARE VERSION:

Sublime Text, 2.0.2+
Atom, 0.177.0
OSX, n/a
Gitbash, n/a
Git, 2.2.1
Python, 2.7/3.6
VirtualBox, 4.2.28
Vagrant, 1.7.2

LENGTH OF PROGRAM*: 5 months
FREQUENCY OF CLASSES: Self-paced
INSTRUCTIONAL TOOLS AVAILABLE: Video lectures, Text instructions, Quizzes, Study Groups, Knowledge, Project Reviews

*This is a self-paced program and the length is an estimation of total hours the average student may take to complete all required coursework, including lecture and project time. Actual hours may vary.

Programming Fundamentals and the Web

Get started as a developer by mastering object-oriented Python programming, HTML, CSS, and responsive Web design.

Supporting Lesson Content: Programming Foundations with Python

Lesson Title	Learning Outcomes
USE FUNCTIONS	• Tour the Python standard library • Use programming library documentation
USE CLASSES: DRAW TURTLES	• Use classes and objects to draw graphics
USE CLASSES: SEND TEXT	• Use the Twilio web API to send SMS messages
USE CLASSES: PROFANITY EDITOR	• Read and write to and from files • Accessing web APIs with the Python urllib library
MAKE CLASSES: MOVIE WEBSITE	• Write programs using Object Oriented Programming (OOP) design
MAKE CLASSES: ADVANCED TOPICS	• Reuse code with class inheritance • Customize inherited classes with method overriding

Project 1: Movie Website

In this project, you will write server-side code to store a list of your favorite movies, including box art imagery and a movie trailer URL. You will then serve this data as a web page allowing visitors to review their movies and watch the trailers.

Supporting Lesson Content: Intro to HTML and CSS

Lesson Title	Learning Outcomes
HTML SYNTAX	• Set up your development environment for writing HTML • Learn about HTML tree structure and write HTML syntax
CSS SYNTAX	• Get started with the basics of CSS • Unleash the power of developer tools to inspect webpages and apply changes on the fly • Use CSS units, colors and fonts • Start adding style to your websites
SIZING	• Use the box model to size and position elements
POSITIONING	• Position elements using different flows
FLOATS	• Use floats to extend and improve your ability to create layouts

Supporting Lesson Content: Responsive Design

Lesson Title	Learning Outcomes
WHY RESPONSIVE?	• Create web pages with mobile-first design • Manage web development by using in-browser development tools • Troubleshoot and debug faulty code
STARTING SMALL	• Build HTML elements for any size screen • Use the browser viewport to create consistent user experiences
BUILDING UP	• Use media queries and breakpoints to create responsive web page designs • Create flexible HTML elements with Flexbox

Project 2: Build a Portfolio

In this project, you will be provided with a portfolio website design mockup as a PDF file and will translate that design into a real website using HTML and CSS.

Developers' Tools

Brush up your knowledge of essential developers' tools such as the Unix shell, Git, and Github; then apply your skills to investigate HTTP, the Web's fundamental protocol.

Supporting Lesson Content: Shell Workshop

Lesson Title	Learning Outcomes
SHELL WORKSHOP	• The Unix shell is a powerful tool for developers of all sorts. You'll get a quick introduction to the very basics of using it on your own computer.

Supporting Lesson Content: Git & GitHub

Lesson Title	Learning Outcomes
PURPOSE & TERMINOLOGY	• Learn about the benefits of version control and install the version control tool Git!
CREATE A GIT REPO	• Create a new repository for your code
REVIEW A REPO'S HISTORY	• Review an existing Git repository's history of commits — the changes that have been made to the project.
ADD COMMITS TO A REPO	• A repository is nothing without commits. In this lesson, you'll learn how to make commits, write descriptive commit messages, and verify the changes you're about to save to the repository.
TAGGING, BRANCHING, AND MERGING	• Being able to work on your project in isolation from other changes will multiply your productivity. You'll learn how to do this isolated development with Git's branches.
UNDOING CHANGES	• Help! Disaster has struck! You don't have to worry, though, because your project is tracked in version control! You'll learn how to undo and modify changes that have been saved to the repository

Supporting Lesson Content: Git & GitHub (Continued)

Lesson Title	Learning Outcomes
WORKING WITH REMOTES	• Create remote repositories on GitHub and send changes to the remote repository
WORKING ON ANOTHER DEVELOPER'S REPOSITORY	• Fork another developer's project and learn how to contribute to a public project
STAYING IN SYNC WITH A REMOTE REPOSITORY	• Send suggested changes to another developer by using pull requests and use the powerful git rebase command to squash commits together

Supporting Lesson Content: HTTP & Web Servers

Lesson Title	Learning Outcomes
REQUESTS & RESPONSES	• Examine HTTP requests and responses by experimenting directly with a web server, interacting with it by hand.
THE WEB FROM PYTHON	• Build up your knowledge of HTTP by writing servers and clients in Python that speak HTTP.
HTTP IN THE REAL WORLD	• Examine a number of practical HTTP features that go beyond basic requests and responses.

The Backend: Databases & Applications

Master SQL databases and build multi-user web applications using the Flask framework, SQLAlchemy, and authentication providers such as Google and Facebook.

Supporting Lesson Content: Intro to Relational Databases

Lesson Title	Learning Outcomes
DATA AND TABLES	• Use the table structure of databases to organize data • Use types and keys to more accurately model your data
ELEMENTS OF SQL	• Use the select statement to retrieve data from tables • Use the insert statement to add data to tables • Combine SQL tables using joins and aggregations to create powerful queries
PYTHON DB-API	• Interact with a database from Python code • Connect a Python web application to an SQL database • Discover and fix security problems with database-backed apps
DEEPER INTO SQL	• Create tables using normalized forms • Use keys to express relationships between tables • Write reusable views to quickly and efficiently retrieve data

Project 3: Logs Analysis

In this project, you'll analyze data from a web service's logs, practicing your command-line and database skills, particularly with a focus on building advanced SQL queries.

Supporting Lesson Content: Intro to Relational Databases

Lesson Title	Learning Outcomes
WORKING WITH CRUD	• Model database entries in Python • Write server code to create, read, update and delete database entries interactively.
MAKING A WEB SERVER	• Configure a web server to handle requests using HTTP • Allow a web server to read and update data based on HTTP request input

Supporting Lesson Content: Intro to Relational Databases (Continued)

Lesson Title	Learning Outcomes
DEVELOPING WITH FRAMEWORKS	• Build a functioning web application using the lightweight Flask Framework • Respond to HTTP requests with JSON dat
ITERATIVE DEVELOPMENT	• Plan the design of a complex web application

Supporting Lesson Content: Authentication and Authorization

Lesson Title	Learning Outcomes
AUTHENTICATION VS AUTHORIZATION	• Scure your application by verifying users' identities • Control application authorization based on user roles and login state • Use third-party systems to authenticate users
CREATING GOOGLE SIGN-IN	• Implement user authentication using Google's OAuth 2.0 tools Local Permission System • Store user data in an application database • Manage user authorization from stored user data
ADDING FACEBOOK AND OTHER PROVIDERS	• Implement other authentication providers in a web app

Supporting Lesson Content: RESTful APIs

Lesson Title	Learning Outcomes
WHAT'S AND WHY'S OF APIS	• Examine API terminology, techniques, and the REST concept.

Supporting Lesson Content: RESTful APIs (Continued)

Lesson Title	Learning Outcomes
ACCESSING PUBLISHED APIS	• Send requests to remote APIs. Use published documentation to understand and apply those APIs correctly.
CREATING YOUR OWN APIS	• Apply the Flask framework to create APIs in Python code.
SECURING YOUR API	• Use token-based authentication and OAuth to protect API endpoints.
WRITING DEVELOPER-FRIENDLY APIS	• Improve your documentation skills and use API versioning to help developers use your API correctly.

Project 4: Build an Item Catalog

In this project, you will develop an application that provides a list of items within a variety of categories as well as provide a user registration and authentication system. Registered users will have the ability to post, edit and delete their own items.

The Frontend: JavaScript and AJAX

Extend the power of the web frontend using JavaScript, JQuery, and AJAX to build advanced interactive web applications.

Supporting Lesson Content: Intro to AJAX

Lesson Title	Learning Outcomes
REQUESTS AND APIS	• Connect to external web APIs to power asynchronous browser updates
BUILDING THE MOVE PLANNER APP	• Use the jQuery Javascript library to build AJAX requests and handle API responses • Handle error responses with AJAX

Supporting Lesson Content: JavaScript Design Patterns

Lesson Title	Learning Outcomes
CHANGING EXPECTATIONS	• React to changing product specifications and developer Expectations • Explore the Model-View-Controller design pattern • Analyze an existing application for MVC structure
REFACTORING WITH SEPARATION OF CONCERNS	• Write code with discrete areas of responsibility in an MVC Application • Refactor an existing application to make use of modern code design practices
USING AN ORGANIZATION LIBRARY	• Build a reactive front end application using an organization library, knockout.js • Implement knockout models and observable elements in an application
LEARNING A NEW CODEBASE	• Use proven strategies to adapt to a new and unfamiliar codebase

Supporting Lesson Content: JavaScript Design Patterns

Lesson Title	Learning Outcomes
GETTING STARTED WITH THE APIS	• Set up your developer credentials and get started with the Google Maps APIs.
UNDERSTANDING API SERVICES	• Explore the location services available in the Google Maps APIs, including the Geocoding, Elevation, and Directions APIs.
USING THE APIS IN PRACTICE	• Examine some technical details of using the Maps APIs.

Project 5: Neighborhood Map

In this project, you will develop a single-page application featuring a map of your neighborhood or a neighborhood you would like to visit. You will then add additional functionality to this application, including: map markers to identify popular locations or places you'd like to visit, a search function to easily discover these locations, and a listview to support simple browsing of all locations. You will then research and implement third-party APIs that provide additional information about each of these locations (such as StreetView images, Wikipedia articles, Yelp reviews, etc).

Deploying to Linux Servers

Supporting Lesson Content: Configuring Linux Web Servers

Lesson Title	Learning Outcomes
INTRO TO LINUX	• Explore the historical roots of Linux and some common Linux Distributions • Launch the Ubuntu operating system in a virtual machine on your own computer
LINUX SECURITY	• Control authorization on a Linux system using super user Privileges • Install additional software packages to a Linux system • Manage Linux users and user permissions • Protect a Linux system with a universal firewall
WEB APPLICATION SERVERS	• Install an Apache web application server on a Linux system

Project 6: Linux Server Configuration

In this project, you will take a baseline installation of a Linux distribution on a virtual machine and prepare it to host your web applications, to include installing updates, securing it from a number of attack vectors, and installing and configuring web and database servers.