WHAT IS FULL STACK WEB DEVELOPER?

Course Content

1

50 Days

Programming Fundamentals and the Web

1. Movie Trailer Website
2. Build a Portfolio Site

2

24 hours

Developers' Tools

- Unix shell
- Git
- GitHub

3

71 Days

The Backend: Databases & Applications

3. Logs Analysis Project
4. Build an Item Catalog Application

4

37 Days

The Frontend: JavaScript & AJAX

5. Neighbourhood Map

5

21 Days

Deploying to Linux Servers

6. Linux Server Configuration

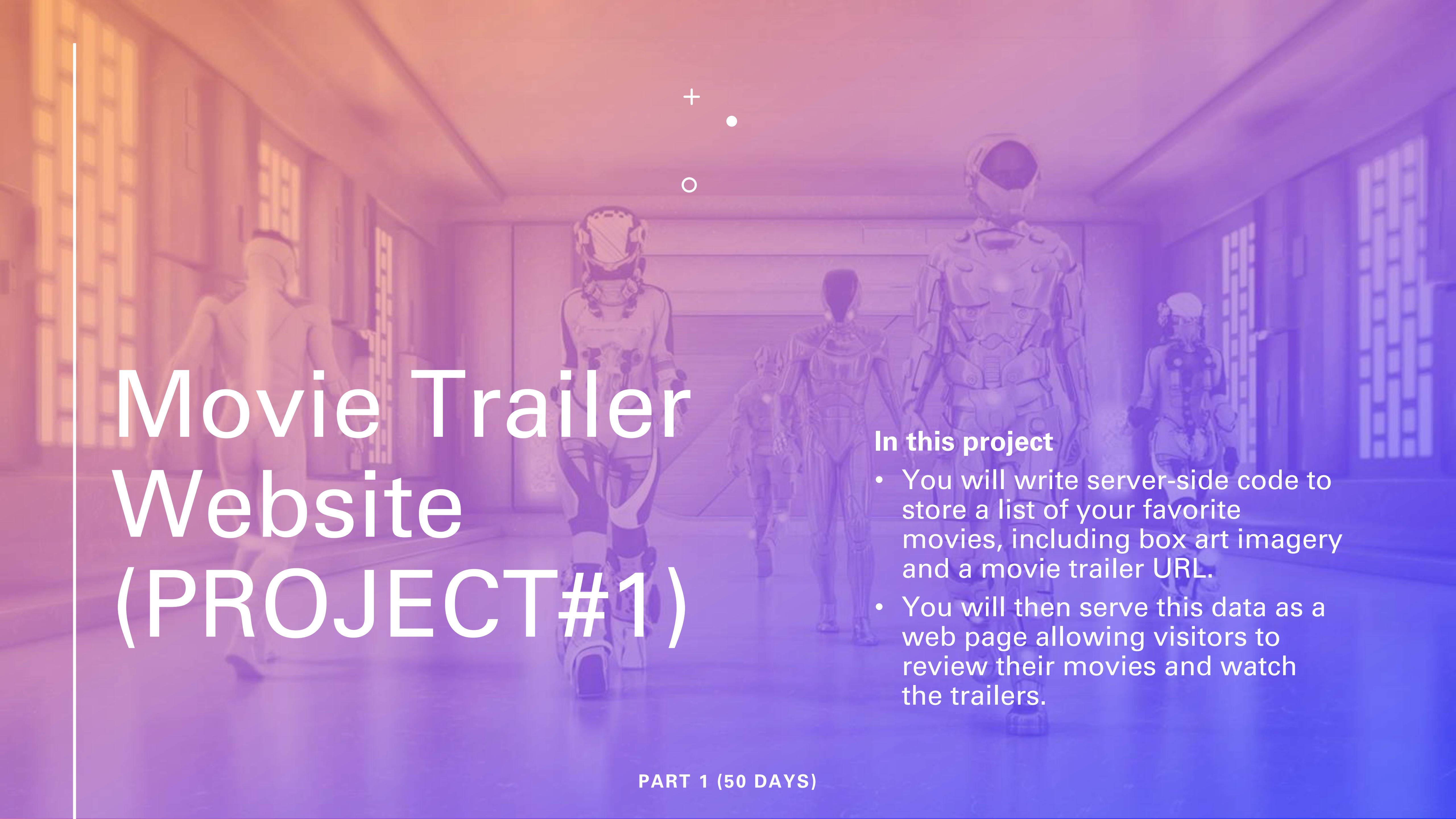
PART 1 (50 DAYS)

Programming Fundamentals and the Web

Programming Foundations with Python

PART 1 (50 DAYS)

1. **Use Functions**
   - Tour the Python standard library
   - Use programming library documentation
2. **Use Classes: Draw Turtles**
   - Use classes and objects to draw graphics
3. **Use Classes: Send Text**
   - Use the Twilio web API to send SMS messages
4. **Use Classes: Profanity Editor**
   - Read and write to and from files
   - Accessing web APIs with the Python urllib library
5. **Make Classes: Movie Website**
   - Write programs using Object Oriented Programming (OOP) design
6. **Make Classes: Advanced Topics**
   - Reuse code with class inheritance
   - Customize inherited classes with method overriding



Movie Trailer Website (PROJECT#1)

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.

PART 1 (50 DAYS)

Intro to HTML and CSS

PART 1 (50 DAYS)

1. **HTML Syntax**
   - Set up your development environment for writing HTML
   - Learn about HTML tree structure and write HTML syntax
2. **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
3. **Sizing**
   - Use the box model to size and position elements
4. **Positioning**
   - Position elements using different flows
5. **Floats**
   - Use floats to extend and improve your ability to create layouts

Responsive Design

1. **Why Responsive?**
   - Create web pages with mobile-first design
   - Manage web development by using in-browser development tools
   - Troubleshoot and debug faulty code
2. **Starting Small**
   - Build HTML elements for any size screen
   - Use the browser viewport to create consistent user experiences
3. **Building Up**
   - Use media queries and breakpoints to create responsive web page designs
   - Create flexible HTML elements with Flexbox

Build a Portfolio (PROJECT#2)

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.

PART 1 (50 DAYS)

PART 2 (1 DAY)

Developers' Tools

Shell Workshop

PART 2 (1 DAY)

Shell Workshop

- The Unix shell is a powerful tool for developers of all sorts. Get a quick introduction to the basics of using it on your computer.

Git & GitHub

PART 2 (1 DAY)

1. Purpose & Terminology
2. Create a Git Repo
3. Review a Repo's History
4. Add Commits to a Repo
5. Tagging, Branching, and Merging
6. Undoing Changes
7. Working with Remotes
8. Working on Another Developer's Repository
9. Staying In Sync With a Remote Repository

HTTP & Web Servers

PART 2 (1 DAY)

1. **Requests & Responses**
   - Examine HTTP requests and responses by experimenting directly with a web server, interacting with it by hand.
2. **The Web from Python**
   - Build up your knowledge of HTTP by writing servers and clients in Python that speak HTTP.
3. **HTTP in the Real World**
   - Examine a number of practical HTTP features that go beyond basic requests and responses.

PART 3 (71 DAYS)

The Backend: Databases & Applications

PART 3 (71 DAYS)

Databases & Applications

1. **Data and Tables**
   - Use the table structure of databases to organize data
   - Use types and keys to more accurately model your data
2. **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
3. **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
4. **Deeper into SQL**
   - Create tables using normalized forms
   - Use keys to express relationships between tables
   - Write reusable views to quickly and efficiently retrieve data

Logs Analysis (PROJECT#3)

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.

PART 3 (71 DAYS)

Full Stack Foundations

PART 3 (71 DAYS)

1. **Working with CRUD**
   - Model database entries in Python
   - Write server code to create, read, update and delete database entries interactively.
2. **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
3. **Developing with Frameworks**
   - Build a functioning web application using the lightweight Flask framework
   - Respond to HTTP requests with JSON data
4. **Iterative Development**
   - Plan the design of a complex web application

Authentication and Authorization

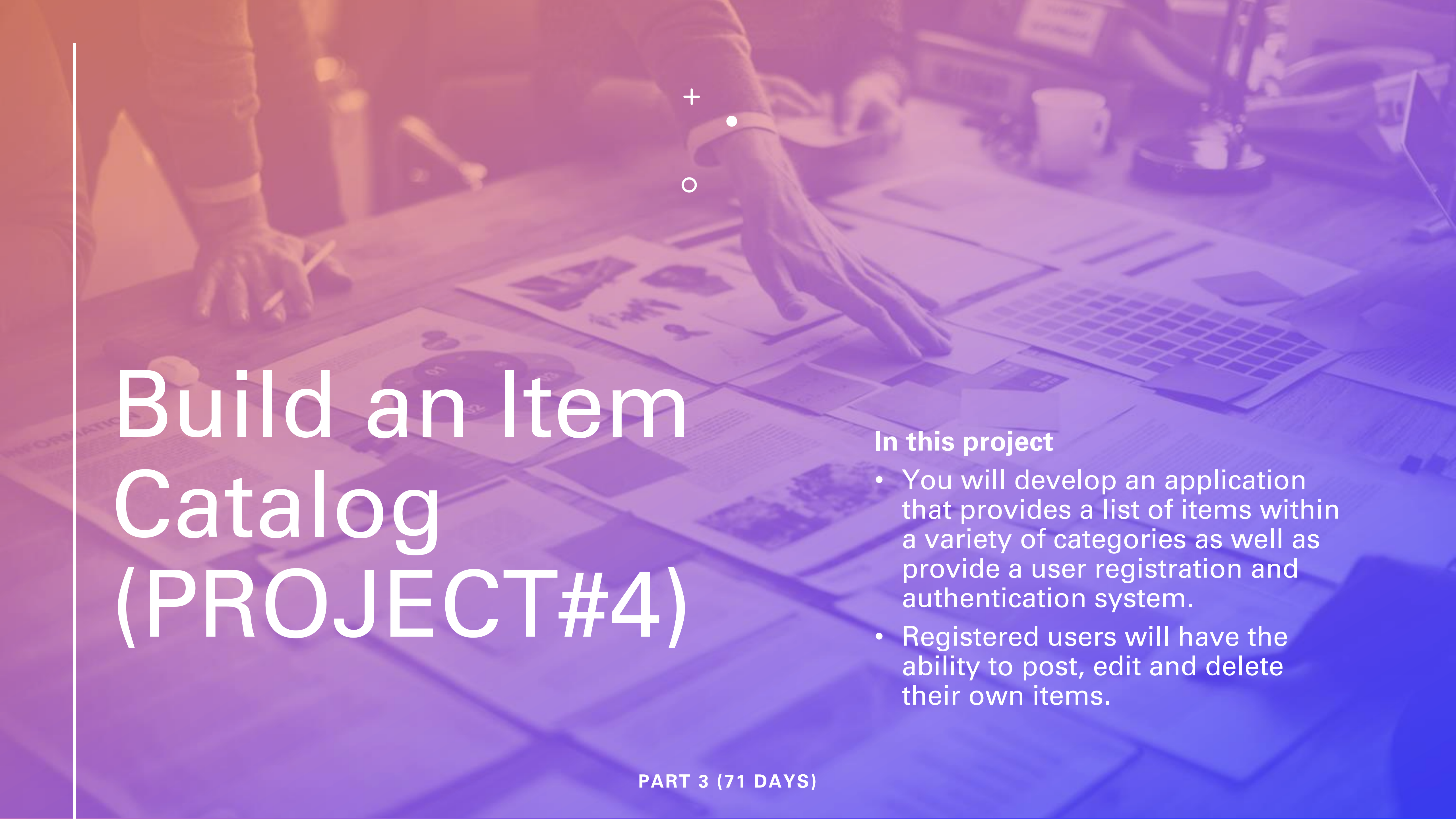
PART 3 (71 DAYS)

1. **Authentication vs Authorization**
    - Secure your application by verifying users' identities
    - Control application authorization based on user roles and login state
    - Use third-party systems to authenticate users
2. **Creating Google Sign-in**
    - Implement user authentication using Google's OAuth 2.0 tools
3. **Local Permission System**
    - Store user data in an application database
    - Manage user authorization from stored user data
4. **Adding Facebook and Other Providers**
    - Implement other authentication providers in a web app

RESTful APIs

PART 3 (71 DAYS)

1. **What's and Why's of APIs**
    - Examine API terminology, techniques, and the REST concept.
2. **Accessing Published APIs**
    - Send requests to remote APIs. Use published documentation to understand and apply those APIs correctly.
3. **Creating Your Own APIs**
    - Apply the Flask framework to create APIs in Python code.
4. **Securing Your API**
    - Use token-based authentication and OAuth to protect API endpoints.
5. **Writing Developer-Friendly APIs**
    - Improve your documentation skills and use API versioning to help developers use your API correctly.



Build an Item Catalog (PROJECT#4)

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.

PART 3 (71 DAYS)

PART 4 (37 DAYS)

The Frontend: JavaScript and AJAX

Intro to AJAX

PART 4 (37 DAYS)

1. **Requests and APIs**
   - Connect to external web APIs to power asynchronous browser updates.
2. **Building the Move Planner App**
   - Use the jQuery JavaScript library to build AJAX requests and handle API responses
   - Handle error responses with AJ

JavaScript Design Patterns

1. **Changing Expectations**
   - React to changing product specifications and developer expectations
   - Explore the Model-View-Controller design pattern
   - Analyze an existing application for MVC structure
2. **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
3. **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
4. **Learning a New Codebase**
   - Use proven strategies to adapt to a new and unfamiliar codebase

Google Maps APIs

1. **Getting Started with the APIs**
   - Set up your developer credentials and get started with the Google Maps APIs.
2. **Understanding API Services**
   - Explore the location services available in the Google Maps APIs, including the Geocoding, Elevation, and Directions APIs.
3. **Using the APIs in Practice**
   - Examine some technical details of using the Maps APIs.

Neighborhood Map (PROJECT#5)

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).

PART 4 (37 DAYS)

PART 5 (21 DAYS)

Deploying to Linux Servers

Configuring Linux Web Servers

PART 5 (21 DAYS)

1. **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
2. **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
3. **Web Application Servers**
   - Install an Apache web application server on a Linux system

Linux Server Configuration (PROJECT#6)

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.

PART 5 (21 DAYS)