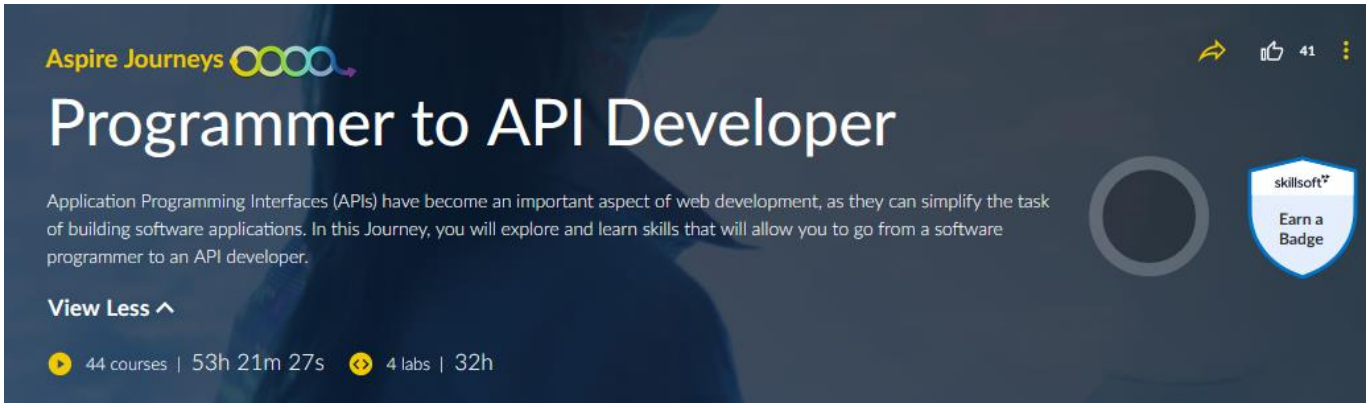
Programmer to API Developer

SKILLSOFT ASPIRE JOURNEY

skillsoft | percipio



Tracks

Track 1: Programmer

In this Skillsoft Aspire track of the Programmer to API Developer journey, the focus is on API development fundamentals, reusable code, refactoring, Agile and TDD best practices for APIs.

Explore 11 courses | 14h 58m 27s 1 lab | 8h

Track 2: Programming Analyst

In this Skillsoft Aspire track of the Programmer to API Developer journey, the focus is on software planning and design, effective Rapid Application Development, automated testing, and penetration ...

View More

Explore 11 courses | 11h 52m 39s 1 lab | 8h

Track 3: Web Services & Systems Programmer

In this Skillsoft Aspire track of the Programmer to API Developer journey, the focus is on API lifecycle and CI/CD best practices, clean and secure coding, API management, and API error handling.

Explore 9 courses | 9h 50m 51s 1 lab | 8h

Track 4: API Developer

In this Skillsoft Aspire track of the Programmer to API Developer journey, the focus is on API end to end development including REST and SOAP, API data models, web technologies, web API security, ...

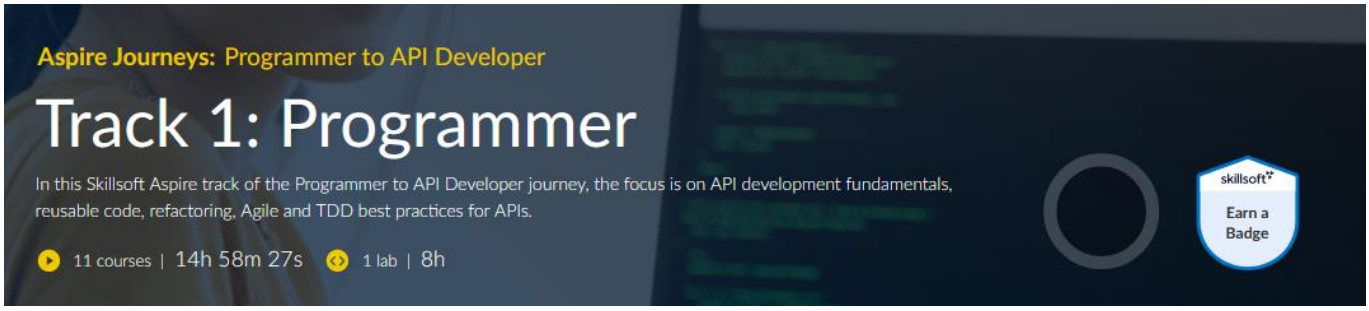
View More

Explore 13 courses | 16h 39m 28s 1 lab | 8h

PREREQUISITES

In order to fully profit from the potential of this Aspire Journey we recommend the following prerequisite skills:

- Knowledge of API development
- Knowledge of software testing
- Familiar with Agile



Patterns in Programming: API Design Patterns

Objectives

- describe the importance of design patterns
- describe what Behavioral design patterns are and their implementations
- create the Iterator design pattern
- describe what creational design patterns are and their implementations
- demonstrate creating the Singleton design pattern
- describe what structural design patterns are and their implementations
- create the Adapter design pattern
- describe and recognize different types of design anti-patterns
- describe how to use code refactoring to implement efficient programming solutions
- describe how to use software refactoring techniques to solve architectural coding problems
- recall the guiding principles that underpin most API patterns
- describe fundamental design patterns for implementing API resource layout and naming

Patterns in Programming: Patterns in API Programming

Objectives

- differentiate between stateless and stateful API services
- describe content negotiation and how it relates to API services
- describe URI templates and how they relate to API services
- describe Design for Intent and how it relates to API services
- describe methods for performing versioning for API services
- describe methods for performing bulk operations using API services
- describe methods for performing pagination with API services
- describe methods for performing filtering and sorting with API services
- describe methods for performing API notification and error logging with API services
- describe the difference between REST and SOAP API web services and when to use each API design pattern
- implement a RESTful web service
- implement a SOAP based web service

Reusable API Code

Objectives

- describe the different types of reusable code patterns
- describe best practices when using reusable code patterns
- describe reusable REST API patterns
- describe how to create reusable code using a multi-tier software architecture process
- describe how to create reusable code using a component-based development process
- describe how to create reusable code using an API First development process
- describe the differences between API documentation, specifications, and definitions
- implement a multi-tier software web application
- implement a component-based library
- differentiate between APIs and web services
- describe reusable REST API web services
- implement a RESTful API reusable web service

Refactoring API Code

Objectives

- describe the need for refactoring code and some best practices for refactoring
- describe the benefits of refactoring code
- describe where code technical debt comes from and how to manage it
- describe when code refactoring makes sense and when it does not
- describe how refactoring relates to test-driven development
- describe available refactoring tools to assist in the refactoring process
- classify the various code refactoring methods and features
- describe the code refactoring methods of coordinating data
- describe the code refactoring methods of simplifying Boolean expressions
- describe the code refactoring methods of simplifying method calls
- describe the code refactoring methods for simplifying class hierarchies
- describe the process of refactoring code to move towards a microservice architecture

Agile & TDD Best Practices for APIs: Test-driven Development Strategy

Objectives

- describe the test-driven development cycle
- demonstrate a test-driven development cycle
- describe the test first development process
- describe test-driven development best practices
- differentiate between test-driven development and test first development
- differentiate between test-driven development and test last development
- describe the acceptance test-driven development process
- differentiate between test-driven development and acceptance test-driven development
- describe the purpose of unit testing and some associated best practices
- describe the characteristics of unit testing and the key qualities of a good unit test
- identify testing tools for performing unit testing
- perform unit testing with frameworks

Agile & TDD Best Practices for APIs: Test-driven Development Design Patterns

Objectives

- describe test-driven development design patterns
- develop code using test-driven development
- describe how to use mock frameworks and mock objects and why they are important
- describe mock frameworks and mock objects and how to incorporate them into test-driven development
- use a mock framework while performing test-driven development
- describe Agile user stories in Agile software development
- describe best practices for using user stories
- describe the purpose of user story mapping as it relates to Agile development
- perform user story mapping with an online tool
- describe how to write scenarios and scenario outlines using Cucumber
- describe how to write scenarios and scenario outlines using the Gherkin scenario syntax
- write scenarios and scenario outlines using Cucumber

Agile & TDD Best Practices for APIs: SOLID Design Principles

Objectives

- describe the SOLID design principles for software development
- demonstrate the Single Responsibility Principle
- demonstrate the Open/Closed Principle
- demonstrate the Liskov Substitution Principle
- demonstrate the Interface Segregation Principle
- demonstrate the Dependency Inversion Principle
- describe how to perform behavior-driven development using Cucumber
- perform behavior-driven development using Cucumber
- describe the behavior-driven development process
- describe the benefits of behavior-driven development for software development
- describe how to integrate behavior-driven development into the Scrum process
- differentiate between test-driven development and behavior-driven development

Agile & TDD Best Practices for APIs: API Microservices Using TDD

Objectives

- differentiate between APIs and microservices
- describe the benefits of using the Microservice Architecture
- describe the process of using test-driven development for creating microservices
- create a simple microservice
- describe how to use test-driven development principles to develop APIs
- describe how to use behavior-driven development principles to develop APIs
- describe types of software testing techniques and how they relate to each other
- describe tools for performing software testing techniques
- perform unit testing
- perform integration testing
- perform regression testing
- perform acceptance testing

API Development: Introduction to APIs

Objectives

- describe the importance of APIs in web development and moving to cloud-based web services
- describe use cases and how they are used for modeling behavior
- create a use case diagram
- describe common API use cases and their differences
- describe common public APIs and web services that can be consumed
- use a weather service web API
- describe common issues and mistakes when developing and using APIs
- describe how to manage the API lifecycle
- describe cloud-based APIs and why they are important to developers
- recognize the important skills required to develop APIs
- describe the characteristics of good APIs
- recognize modern trends when designing APIs

API Development: Framework Security

Objectives

- describe API security best practices for REST and SOAP APIs
- compare the security differences of REST and SOAP APIs
- describe the best security practices for REST APIs
- describe the best security practices for SOAP APIs
- describe API development frameworks and when to use them
- recognize how to work with the ASP.NET Web API framework
- describe JAVA API frameworks and their components
- describe PHP API frameworks and their components
- describe Node.js API frameworks and their components
- describe Python API frameworks and their components
- recognize how to work with the JAVA REST API framework
- recognize how to work with the PHP REST API framework

Final Exam: Programmer

Objectives

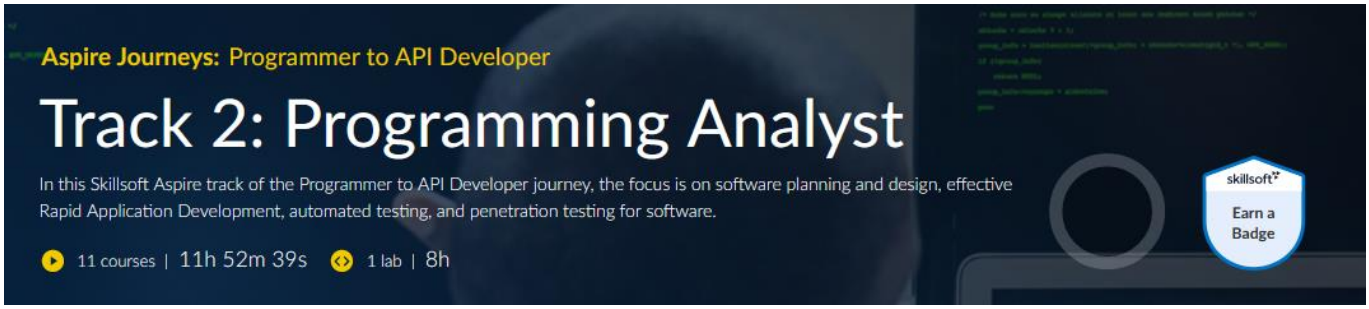
- compare the differences between APIs and microservices
- compare the security differences between REST and SOAP APIs
- demonstrate creating the Singleton design pattern
- demonstrate how to create a Use Case diagram
- demonstrate how to implement a multi-tier software web application
- demonstrate how to implement a SOAP-based web service
- demonstrate the ASP.Net Web API Framework
- demonstrate the Open/Closed Principle
- demonstrate the Single Responsibility Principle
- describe Agile user stories in Agile software development
- describe API development frameworks and when to use each
- describe API Security best practices for REST and SOAP APIs
- describe common public APIs and web services that can be consumed
- describe content negotiation and how it relates to API services
- describe how refactoring relates to test-driven development
- describe how to create reusable code using a component-based development process
- describe how to create reusable code using a multi-tier software architecture process
- describe how to manage the API Lifecycle
- describe how to perform Behavior Driven Development using Cucumber
- describe how to use software refactoring techniques to solve architectural coding problems
- describe how to use Test-driven Development principles to develop APIs
- describe issues and mistakes when developing and using APIs
- describe methods for performing Bulk Operations using API services
- describe methods for performing pagination with API services
- describe methods for performing versioning for API services
- describe Mock frameworks and Mock objects and how to incorporate them into Test-driven Development
- describe test-driven development best practices
- describe Test-driven Development design patterns
- describe the Behavior Driven Development process
- describe the benefits of Behavior Driven Development for software development
- describe the benefits of refactoring code
- describe the benefits of using the Microservice Architecture
- describe the best practices for using user stories
- describe the best practices when using reusable code patterns
- describe the best security practices for REST APIs

- describe the best security practices for SOAP APIs
- describe the code refactoring methods of simplifying Boolean expressions
- describe the difference between API documentation, specifications, and definitions
- describe the differences between stateless and stateful API services
- describe the different types of reusable code patterns
- describe the importance of API in web development and moving to Cloud-based web services
- describe the importance of design patterns
- describe the need for refactoring code and some best practices for refactoring
- describe the process of using Test-driven Development for creating Microservices
- describe the purpose of user story mapping as it relates to Agile development
- describe the SOLID design principles for software development
- describe the test-driven development cycle
- describe the test-first development process
- describe tools for performing software testing techniques
- describe Use Cases and how they are used for modeling behavior
- describe using Mock frameworks and Mock objects and why they are important
- describe what Behavioral design patterns are and their implementations
- describe what creational design patterns are and their implementations
- describe what structural design patterns are and their implementations
- describe when code refactoring makes sense and when it does not
- describe where technical code debt comes from and how to manage it
- differentiate between test-driven development and acceptance test-driven development
- differentiate between test-driven development and test-first development
- differentiate between test-driven development and test last development
- recognize the process of using Test-driven Development for creating Microservices

Programmer

Objectives

- In this lab, you will perform programmer tasks such as implementing the Iterator design pattern, creating a RESTful API Web Service and a SOAP based web service, and creating a multi-tier web application. Then, test your skills by answering assessment questions after using refactoring tools, creating code using SOLID design principles, creating a simple microservice and creating unit tests.



Human-centered Software Design

Objectives

- describe the key concepts of human-centered design
- describe the main principles of human-centered design
- describe the benefits of human-centered design
- describe the best practices of human-centered design
- describe the some bad practices of human-centered design and how to prevent these practices
- describe the concept of human-centered design and how if differs from traditional design
- discover how human-centered design is used by reviewing examples
- describe how qualitative data coding is used to apply data gathered from research
- describe how to formulate hypotheses based on the results of qualitative data coding
- differentiate between storyboarding and prototyping
- describe common methods and best practices for prototyping
- describe how to evaluate prototypes and make data-informed decisions

Software Requirements Planning

Objectives

- recognize the key elements that make up the requirements engineering process
- recognize the appropriate types of stakeholders to help determine software requirements
- recognize methods for gathering stakeholder demands and turning them into requirements
- recognize how to coach stakeholders on possible challenges with some of their requirements and help them choose the right path
- recognize techniques for identifying requirements
- recognize best practices on how to turn requirements into software specifications that are complete, concise, correct, consistent, testable, and unambiguous
- recognize techniques to help prioritize features, and determine the MVP and what can be delayed to future sprints
- recognize tips and tricks to identify hidden requirements
- recognize tips and tricks to create a requirements traceability matrix
- recognize how to approach requirements planning for API development
- demonstrate the effective use of requirements engineering applied to software development lifecycles
- demonstrate how to validate software requirements

Effective RAD: Rapid Application Development

Objectives

- describe the key concepts of Rapid Application Development
- describe the Rapid Application Development Model and its advantages and disadvantages
- describe the traditional Waterfall System Development Life Cycle
- differentiate between Rapid Application Development and the Traditional Waterfall System Development Life Cycle
- differentiate between Rapid Application Development and the Agile framework
- describe the considerations and characteristics of Rapid Application Development
- describe the best practices when performing rapid software delivery
- describe the customer-oriented Rapid Application Development framework
- describe the best practices for creating a customer-focused development culture
- describe the tools and frameworks that can be used for Rapid Application Development
- describe the tools and platforms that can be used for cross-platform Rapid Application Development
- quickly create an application using the Xamarin development tool

Effective RAD: Agile Rapid Application Development

Objectives

- describe the principles of the Agile software process
- describe how the principles of Agile development can be applied to Rapid Application Development
- describe how to manage quality during the Agile software process
- describe quality concerns that can occur during the Agile software process
- describe rapid software prototyping and its benefits
- describe the pros and cons of rapid prototyping
- describe the steps to achieving effective prototyping
- describe how the CI/CD pipeline can be used to deliver software at a higher velocity
- describe how continuous integration can be used to deliver software at a higher velocity
- describe how continuous delivery can be used to deliver software at a higher velocity
- describe how continuous deployment can be used to deliver software at a higher velocity
- describe the differences between continuous integration, delivery, and deployment

Automated Testing: Design Patterns

Objectives

- describe the importance of automated testing and some of its benefits
- describe the roles and responsibilities of the software developer when performing automated testing
- describe how unit tests can be effectively used in automated testing
- describe how automation testing can be used to perform functional testing
- compare API testing with traditional software testing features
- describe how design patterns are effectively used in test automation
- describe some of the common design patterns for testing applications such as command line, GUI, and web applications
- describe tools and frameworks that can be used in API testing
- navigate with the Selenium WebDriver
- create a unit test using a Selenium page object
- construct a Selenium page object for a test case
- implement automated web testing using Selenium unit tests

Pen Testing for Software Development: The Penetration Testing Process

Objectives

- describe what penetration testing is
- recognize the importance of penetration testing
- describe what system hardening is
- identify requirements of penetration testing
- differentiate between penetration testing and vulnerability assessments
- identify types of penetration testing
- identify the stages of penetration testing
- list the different methods of penetration testing
- recognize the differences between white box, black box, and gray box penetration testing
- describe the differences in penetration testing methodologies
- identify the tools available for penetration testing
- recognize the types of outputs of penetration testing
- identify penetration testing best practices
- perform a penetration test

Pen Testing for Software Development: Penetration Testing SDLC, Team Structure, & Web Services

Objectives

- recognize where penetration testing falls within the software development life cycle
- differentiate between penetration testing and development
- identify the importance of developer contributions to penetration testing
- identify a penetration testing team structure
- describe the tasks of the penetration testing blue team
- describe the tasks of the penetration testing red team
- describe the tasks of the penetration testing purple team
- describe the importance of performing penetration tests on web services and APIs due to their vulnerabilities
- identify what is involved in performing penetration tests on APIs
- recognize the tools available for API penetration testing
- perform a penetration test on a REST API
- perform a penetration test using Burp Suite

API Design: API Development

Objectives

- describe what needs to be considered during API development
- describe constraints that can impact REST API development
- describe different methods and programming languages for consuming REST API web services
- describe constraints that can impact SOAP API development
- describe the best practices to take when creating SOAP web services
- describe security threats and risks to consider when creating web APIs
- describe methods of mitigating security risks when creating web APIs
- describe the importance of creating code that is reusable
- describe methods for creating API code that is reusable
- create a REST API web service
- create a SOAP web service
- reuse code using the Decorator pattern

API Design: API Strategy & Design

Objectives

- describe strategies for designing API web services
- describe the use of Richardson Maturity Model to identify the design maturity of REST APIs
- describe how to design APIs with reusability in mind
- describe the best practices for naming and versioning API URIs
- describe the tools available with Swagger including the API editor, the API code and client generator, the REST API documentation tool, and the REST API testing tool
- describe the benefits and best practices of using the OpenAPI specification standards
- describe how to use OpenAPI to create REST APIs
- describe how to create REST APIs using YAML
- describe tools that can be used to manage APIs
- create an OpenAPI definition with the Swagger Editor using YAML
- describe the use of Swagger's interactive web API help pages
- describe the API management features of a Swagger-generated server

API Design: RESTful APIs

Objectives

- describe best practices to follow when developing and designing REST APIs
- describe how REST APIs are used by reviewing real world examples
- describe best practices for generating REST API document and review examples
- describe best design practices for working with REST API resources
- describe best design practices for working with REST API CRUD implementations
- describe best design practices for working with REST API error handling
- describe best design practices for working with REST API HTTP status codes
- describe best design practices for working with REST API change management and versioning
- describe best design practices for working with REST API pagination
- describe best design practices for working with REST API partial responses
- describe best practices for working with JSON Web Tokens
- describe best practices for working with API keys

Programming Analyst

Objectives

- In this lab, you will perform programming Analyst tasks such as creating prototypes using the Pencil Project, creating an application using a rapid application development tool, implementing a CI/CD pipeline and testing a web application with Selenium Web Driver. Then, test your skills by answering assessment questions after creating SOAP and REST API web services, writing YAML using Swagger Editor and handling REST API errors.

Final Exam: Programming Analyst

Objectives

- compare API testing with traditional software testing features
- compare Rapid Application Development with the Agile framework
- compare Rapid Application Development with the Traditional Waterfall Development Live Cycle
- compare the differences between storyboarding and prototyping
- create a REST API web service
- describe best design practices for working with REST API error handling
- describe best design practices for working with REST API partial responses
- describe best design practices for working with REST API resources
- describe best practice for generating REST API document and review examples
- describe constraints that can impact REST API development
- describe different methods and programming languages for consuming REST API web services
- describe how automation testing can be used to perform functional testing

- describe how Continuous Delivery can be used to deliver software at a higher velocity
- describe how Continuous Deployment can be used to deliver software at a higher velocity
- describe how Continuous Integration can be used to deliver software at a higher velocity
- describe how design patterns are effectively used in test automation
- describe how REST APIs are used by reviewing real-world examples
- describe how the CI/CD Pipeline can be used to deliver software at a higher velocity
- describe how to use OpenAPI to create REST APIs
- describe how unit tests can be effectively used in automated testing
- describe methods for creating reusable API code
- describe some bad practices of Human-centered design and how to prevent these practices
- describe strategies for designing API web services
- describe the best practices for naming and versioning API URIs
- describe the best practices of human-centered design
- describe the best practices to follow when developing and designing REST APIs
- describe the concept of human-centered design and how it differs from traditional design
- describe the considerations and characteristics of Rapid Application Development
- describe the differences between Continuous Integration, Delivery, and Deployment
- describe the importance of automated testing and some of its benefits
- describe the key concepts of human-centered design
- describe the key concepts of Rapid Application Development
- describe the main principles of human-centered design
- describe the principles of the Agile software process
- describe the Rapid Application Development Model and its advantages and disadvantages
- describe the Richardson Maturity Model to identify the design maturity of REST APIs
- describe the roles and responsibilities of the software developer when performing automated testing
- describe the tasks of the penetration testing blue team
- describe the tools available with Swagger including the API editor, the API code and client generator, the REST API documentation tool, and the REST API testing tool
- describe the traditional Waterfall System Development Life Cycle
- describe what needs to be considered during API development
- identify a penetration testing team structure
- identify the appropriate tool used for developing the RESTful APIs
- identify the importance of developer contributions to penetration testing
- identify the stages of penetration testing
- identify types of penetration testing
- recognize best practices on how to turn requirements into software specifications that are complete, concise, correct, consistent, testable, and unambiguous
- recognize methods for gathering stakeholder demands and turning them into requirements
- recognize techniques for identifying requirements
- recognize the appropriate types of stakeholders to help determine software requirements
- recognize the differences between White box, Black box, and Grey box penetration testing
- recognize the importance of penetration testing
- recognize the key elements that make up the requirements engineering process
- recognize tips and tricks to identify hidden requirements
- recognize where penetration testing falls within the software development lifecycle (SDLC)
- reuse code using the Decorator pattern
- use OpenAPI to create REST APIs

API CI/CD Best Practices: API Lifecycle Management

Objectives

- describe the phases of the API Lifecycle Management process
- describe the benefits of the API Lifecycle Management process
- describe the Producer and Consumer design pattern
- describe API strategy and planning
- describe API implementation and operation
- describe API governance, associated best practices, and models that can be used when implementing API governance
- describe API Lifecycle Management best practices, such as monitoring, analytics, security, and versioning techniques
- design a Swagger REST API
- implement a Swagger REST API server
- test a Swagger REST API
- implement a Swagger REST API client

API CI/CD Best Practices: API Continuous Integration/Continuous Deployment

Objectives

- describe continuous integration and deployment and how to choose the correct strategy
- describe the benefits of using continuous integration and continuous deployment
- describe CI/CD principles relating to deploying APIs
- describe how to test APIs during continuous integration and best practices when performing API testing
- describe how to use CI/CD pipelines and how they apply to APIs
- describe how to scale the CI/CD pipeline
- describe CI/CD best practices
- recognize popular CI/CD tools, such as Jenkins, TeamCity, CircleCI, Travis CI, GitHub, and GitLab
- set up GitHub and AWS environments for CI/CD use
- install Jenkins on AWS EC2, build Swagger server from a GitHub repository, and deploy to AWS Elastic Beanstalk
- integrate Travis CI with a GitHub repository, setup the pipeline to build Swagger server, and deploy to AWS Elastic Beanstalk
- set up Actions on a GitHub repository, build Swagger server, and deploy to AWS Elastic Beanstalk

API Strategy Design & Implementation: API Strategy Best Practices

Objectives

- describe the API strategy mindset and the importance of crafting this strategy
- outline the essentials and best practices of API strategies
- outline the phases of the API life cycle and how to manage it
- identify the challenges faced when creating an API strategy
- recognize the threats and risks that need to be considered when creating an API strategy
- identify the advantages of creating an API strategy
- describe to crucial elements to consider when developing an API strategy
- identify API standards that have been created by governments and the value of these standards
- recognize the API governance framework as a vital building block for security
- compare the advantages and challenges of building versus buying API libraries
- distinguish the benefits and challenges of using either a private, partner, or public API when developing your API strategy
- list real-world examples of using and creating API strategies

API Clean and Secure Coding: Clean Coding

Objectives

- describe the importance of writing clean code and the challenges of keeping code clean
- describe the qualities of clean code
- describe best practices for writing clean code
- describe the benefits of writing clean code
- describe coding bad habits and how to prevent them
- describe the business cost of writing messy, disorganized code
- describe methods for refactoring and cleaning up messy code
- describe best practices for commenting code
- describe best practices for writing clean understandable API code
- differentiate between clean and dirty code
- describe tools that will help you write clean code
- describe tools that can be used to clean web code such as HTML, CSS, and JavaScript

API Clean and Secure Coding: API Secure Coding

Objectives

- outline how to write secure API code
- recognize best practices when writing secure API code
- list the features that should be present when developing API code
- identify tools that can be used for API development
- describe Web API security and how to ensure it's done correctly
- recognize best practices for ensuring API security
- identify REST API security principles and outline tips to provide better security
- outline the purpose of OAuth and how it is used to perform single sign-on (SSO)
- describe various methods for performing API authorization
- list the different OAuth grant types and identify when to use each grant type
- define SAML and OAuth and distinguish how they're used to achieve single sign-on, federation, and identity management
- define Transport Layer Security and identify its strengths and weaknesses

API Management: Strategy & Monitoring

Objectives

- describe the key characteristics of API management strategies
- describe best practices for defining API management strategies
- describe the key characteristics of API monitoring strategies
- describe best practices for defining API monitoring strategies
- describe the features and benefits of API management tools
- use the Kong Enterprise tool to for API management
- describe the API monitoring tools and their strengths and weaknesses
- use the Assertible API monitoring tool
- recognize some of the best metrics monitoring APIs
- describe the benefits of API testing and using SoapUI automation
- perform automated API testing using SoapUI
- describe the roles of containers and API microservices and how they can help your API monitoring strategy

API Management: Elasticsearch API Analytics

Objectives

- describe the components of the ELK Stack and how they work together
- describe how to perform API analysis using the ELK Stack
- describe the purpose and benefits of Elasticsearch
- describe the purpose and benefits of Kibana
- describe the purpose and benefits of Beats
- describe the purpose and benefits of Logstash
- install and configure Elasticsearch
- install and configure Kibana
- install and configure Elasticsearch Beats
- install and configure Logstash
- describe how to use Elasticsearch Notifications to notify staff when the API services have issues
- install and configure Elasticsearch Notifications

API Error Handling: Best Practices

Objectives

- describe how to handle API error responses and identify the response messages
- describe the sub-elements and fault codes that are part of the SOAP fault block
- describe some of the best practices of API error handling
- describe the types of HTTP response codes
- describe the available REST API response standards and the benefits and weaknesses of each standard
- describe API error messages that are returned from API method calls for large corporations
- describe the Facebook Graph API web service and the components, error handling features, and functionality
- use HTTP status codes to build a REST service in .NET Core
- describe how to provide default .NET Core error responses
- return basic responses in .NET Core by providing appropriate response codes
- describe how to provide detailed responses using custom errors using .NET Core
- create and use custom errors using .NET Core

Final Exam: Web Services & Systems Programmer

Objectives

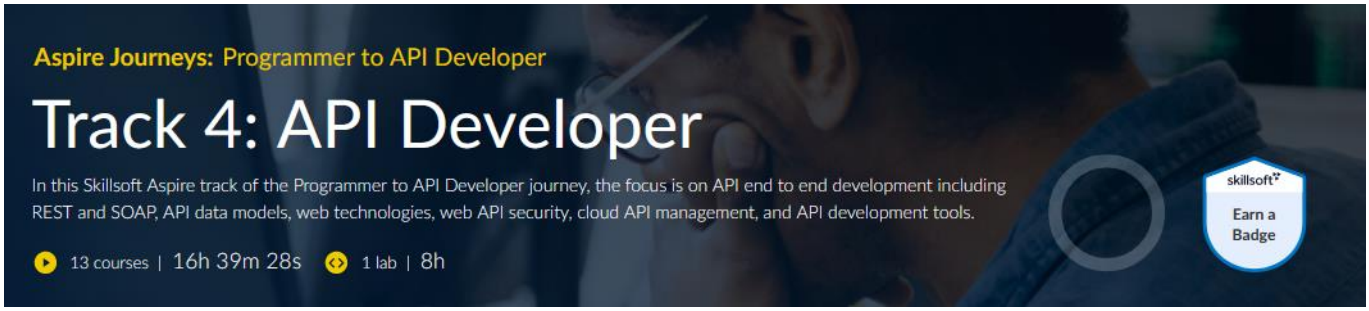
- demonstrate how to create and use custom errors using Spring
- demonstrate how to install and configure Elasticsearch
- demonstrate how to install and configure Elasticsearch Beats
- demonstrate how to install and configure Kibana
- demonstrate how to install and configure Logtash
- demonstrate how to use HTTP status codes to build a REST service in Spring
- describe API error messages which are returned from API method calls for large corporations
- describe API governance, associated best practices, and models that can be used when implementing API governance
- describe API implementation and operation
- describe API Lifecycle Management best practices, such as monitoring, analytics, security, and versioning techniques
- describe API strategy and planning
- describe bad coding habits and how to prevent using them
- describe best practices for defining API management strategies
- describe best practices for defining API monitoring strategies
- describe best practices for ensuring API security
- describe CI/CD best practices
- describe CI/CD principles relating to deploying APIs
- describe continuous integration and deployment and how to choose the correct strategy
- describe how to handle API error responses and identify the response messages
- describe how to perform API analysis using the ELK Stack
- describe how to scale the CI/CD pipeline
- describe how to write secure code when writing API code
- describe REST API security principles and tips to ensure better security
- describe SAML and OAuth and how they are used to achieve single sign-on, federation and identity management
- describe some of the best practices of API error handling
- describe the advantages of creating an API strategy
- describe the API monitoring tools and their strengths and weaknesses
- describe the API strategy mindset and the importance of crafting this strategy
- describe the available REST API response standards and the benefits and weaknesses of each standard
- describe the benefits of API testing and using SoapUI automation
- describe the benefits of the API Lifecycle Management process
- describe the benefits of using continuous integration and continuous deployment
- describe the benefits of writing clean code
- describe the best practices for commenting code
- describe the best practices for writing clean code
- describe the business cost of writing messy, disorganized code
- describe the components on the ELK Stack and how it works together
- describe the essentials and best practices of API strategies
- describe the features and benefits of API management tools
- describe the importance of writing clean code and the challenges of keeping code clean
- describe the key characteristics of API management strategies
- describe the key characteristics of API monitoring strategies
- describe the phases of the API life cycle and how to manage it
- describe the phases of the API Lifecycle Management process
- describe the Producer and Consumer design pattern
- describe the purpose and benefits of Elasticsearch
- describe the purpose and benefits of Kibana
- describe the purpose of OAuth and how it is used to perform single sign-on (SSO)
- describe the qualities of clean code
- describe the sub-elements and fault codes which are part of the SOAP fault block

- describe the types of HTTP response codes and explain on each code
- describe threats and risks that need to be considered when creating an API strategy
- describe what Transport Layer Security and its strengths and weaknesses is
- describe what WEB API security and how to ensure it is done correctly is
- identify continuous integration and deployment and how to choose the correct strategy
- identify the phases of the API life cycle and how to manage it
- recognize the purpose of OAuth and how it is used to perform single sign-on (SSO)
- use Gitlab CI

Web Services and Systems Programmer

Objectives

- In this lab, you will perform Web Services and Systems Programmer tasks such as implementing a Swagger REST API server, implementing a CI/CD pipeline and creating clean code using CSS libraries. Then, test your skills by answering assessment questions after implementing SSO using OAuth2, connecting to a SOAP web service and performing testing, installing and configuring the ELK stack, and handling REST API web service errors.



API Development: REST & SOAP Web Services

Objectives

- describe the benefits and constraints of using web services as part of your business strategy
- demonstrate how to create a traditional SOAP web service using Visual Studio 2019
- demonstrate how to consume a traditional SOAP web service using SoapUI
- illustrate how to create a REST API web service using Visual Studio 2019
- demonstrate how to consume a Nasa REST API web service using SoapUI
- describe the relationship between URLs and URIs and when to use each
- illustrate how to perform Web API routing using Visual Studio with convention-based routing
- show how to implement Web API attribute routing in Visual Studio
- describe schema-first design using an API specification language
- create a REST API using the OpenAPI language with the Swagger Editor to generate source code

API Development: REST API Data Models

Objectives

- describe the advantages and disadvantages of using either XML or JSON for transferring and receiving data from web services
- demonstrate how to perform JSON serialization in Web API
- illustrate how to perform XML serialization in Web API
- describe JSON and how it can be used with REST APIs
- demonstrate how to create a JSON model for Web API
- show how to perform model validation in Web API
- classify the advantages and disadvantages of using either private or public APIs
- illustrate how to connect to REST APIs using WebRequest
- show how to connect to REST APIs using HttpClient in .Net Core
- simulate how to create an API gateway to hide underlying private API calls

API Development: Web API Technologies

Objectives

- describe the various standards for returning data from a web service
- demonstrate how to implement a standard wrapper around JSON responses
- demonstrate how to serialize data as XML responses
- show how to perform create, retrieve, update, delete, associate and disassociate Common Data Service entity records
- demonstrate how to use OData V4 query syntax and functions and Common Data Service query functions
- illustrate how to perform conditional operations with ETag criteria
- show how to use bound and unbound functions and actions as well as custom actions
- demonstrate how to implement API route names and make effective use of nouns and verbs
- illustrate how to deploy an API project to a local server
- show how to deploy a web application project to the cloud

API Development: Client-side Web Service Consumption

Objectives

- describe various methods for consuming web services from client-side devices
- illustrate how to consume a REST API service using AngularJS
- demonstrate how to consume a REST API service using ReactJS
- illustrate how to consume a REST API service using Vue.js
- demonstrate how to consume a REST API service using JavaScript
- illustrate how to consume a REST API service using jQuery
- demonstrate how to consume a REST API service using TypeScript
- illustrate how to consume a SOAP web service using JavaScript
- demonstrate how to consume a SOAP web service using jQuery
- illustrate how to consume a SOAP web service using Node.js

API Development: REST API Semantics

Objectives

- describe best practices for defining API schemas and using verbs and nouns
- illustrate how to implement a GET method using a REST API web service
- illustrate how to implement a POST method using a REST API web service
- illustrate how to implement a PUT method using a REST API web service
- illustrate how to implement a DELETE method using a REST API web service
- demonstrate how to consume a GET method using a REST API web service
- demonstrate how to consume a POST method using a REST API web service
- demonstrate how to consume a PUT method using a REST API web service
- demonstrate how to consume a DELETE method using a REST API web service
- illustrate how to use API Route constraints

API Development: HTML5 & Hypermedia

Objectives

- describe what is meant by Hypertext and Hypermedia and distinguish their differences
- implement custom media types in ASP.Net Core
- demonstrate HTML5 input types using examples
- demonstrate the features of HTML5 Canvas using examples
- demonstrate the HTML5 SVG graphics features using examples
- demonstrate the HTML5 audio features using examples
- demonstrate the HTML5 video features using examples
- demonstrate the HTML5 web storage features using examples
- demonstrate the HTML5 web worker features using examples
- demonstrate the HTML5 geolocation features using examples

API Development: URIs & Caching

Objectives

- describe best practices when defining URIs
- perform API versioning using URIs
- create a URI object using the UriBuilder class
- demonstrate how URI templates are used to provide guidelines for developers
- describe best practices for implementing API resources
- use the Windows Workflow Foundation to create a set of activities
- implement in-memory caching using ASP.Net Core
- implement distributed caching using ASP.Net Core
- implement caching using response caching middleware in ASP.Net Core
- perform advanced REST client testing using the Chrome browser

API Development: Web API Security

Objectives

- recognize the functions and characteristics of OAuth and API Security
- authenticate an API using local logins
- authenticate an API using an external authorization service
- illustrate how to prevent Cross Site Request Forgery (CSRF) attacks
- enable a cross-origin request in Web API 2
- illustrate how to use filters in Web API
- implement basic authentication in Web API
- implement forms authentication in Web API
- implement Windows authentication in Web API
- enforce SSL in a Web API controller

API Development: Firebase Backend as a Service

Objectives

- recognize the benefits of using a Cloud-based back-end service such as Firebase
- manage Firebase projects using the Firebase Management REST API
- use Firebase Authentication to add FirebaseUI authentication to your web application
- perform CRUD operations using the Firebase Realtime Database REST API
- configure Firebase's Cloud Firestore NoSQL database
- configure the Firebase Cloud Storage service and use it to perform related operations
- configure Firebase Cloud Functions and add and consume custom functions
- configure Firebase Hosting to store and consume static assets
- describe the features of the Firebase Machine Learning Kit
- implement Firebase Performance Monitoring on a web app

API Development: Cloud API Management

Objectives

- describe how Azure API Management can be used for deploying and hosting API web services
- import and publish APIs using Azure API Management
- create and publish API products using Azure API Management
- create mock API responses using Azure API Management
- transform and protect APIs using Azure API Management
- monitor published APIs using Azure API Management
- debug APIs using Azure API Management
- create non-breaking revision changes using Azure API Management
- publish multiple API versions using Azure API Management
- demonstrate how to use the Azure API Management developer portal

API Development: Cloud API Gateways

Objectives

- describe the characteristics, goals, and various use cases for API gateways in REST API web service deployment
- demonstrate how to develop an HTTP API using the API gateway
- demonstrate how to publish an HTTP API using the API gateway
- demonstrate how to maintain an HTTP API using the API gateway
- illustrate how to develop a REST API using the API gateway
- illustrate how to publish a REST API using the API gateway
- illustrate how to maintain a REST API using the API gateway
- demonstrate how to develop a WebSocket API using the API gateway
- demonstrate how to publish a WebSocket API using the API gateway
- demonstrate how to maintain a WebSocket API using the API gateway

API Development: Tools

Objectives

- download and install the Node.js web server
- download and install the Apache Tomcat web server
- download and install the Windows IIS web server
- download and install the MySQL database server
- download and install the MongoDB database server
- download and install Visual Studio for web and API development
- download and install Visual Code for web and API development
- download and install Eclipse for web and API development
- download and install IntelliJ IDEA for web and API development
- download and install the Docker Desktop virtualization environment

Final Exam: API Developer

Objectives

- demonstrate basic authentication in Web API
- demonstrate forms authentication in Web API
- demonstrate how to authenticate an API using local logins
- demonstrate how to configure the Cloud Firestore NoSQL database
- demonstrate how to configure the Cloud Realtime database
- demonstrate how to connect to REST APIs using HttpClient in .Net Core
- demonstrate how to consume a traditional SOAP web service using SoapUI
- demonstrate how to create a JSON model for Web API
- demonstrate how to create a traditional SOAP web service using Visual Studio 2019
- demonstrate how to create Mock API responses using Azure API Management
- demonstrate how to create non-breaking revision changes using Azure API Management
- demonstrate how to debug APIs using Azure API Management
- demonstrate how to develop an HTTP API using the API Gateway
- demonstrate how to developer a REST API using the API Gateway
- demonstrate how to developer a REST using the API Gateway
- demonstrate how to download and install Eclipse for web and API development
- demonstrate how to download and install the MongoDB database server
- demonstrate how to download and install the MySQL database server
- demonstrate how to download and install the NodeJS webserver
- demonstrate how to implement a DELETE method using a REST API web service
- demonstrate how to implement a GET method using a REST API web service
- demonstrate how to implement a POST method using a REST API web service
- demonstrate how to implement a PUT method using a REST API web service
- demonstrate how to implement a standard wrapper around JSON responses
- demonstrate how to implement Firebase Authentication on a webpage
- demonstrate how to manage Firebase using the REST API
- demonstrate how to monitor published APIs using Azure API Management
- demonstrate how to perform conditional operations you with Etag criteria
- demonstrate how to perform create, retrieve, update, delete, associate and disassociate Common Data Service entity records
- demonstrate how to prevent Cross-Site Request Forgery (CSRF) attacks
- demonstrate how to publish an HTTP API using the API Gateway
- demonstrate how to serialize data as XML responses
- demonstrate HTML5 SVG graphic features with examples
- demonstrate install Eclipse for web and API development
- demonstrate the HTML5 Audio features with examples
- demonstrate the HTML5 Canvas features with examples
- demonstrate the HTML5 input types with examples
- describe how Azure API Management can be used for deploying and hosting API web services
- describe how to consume a REST API service using JavaScript
- describe how to consume a REST API service using jQuery

- describe how to consume a REST API service using Typescript
- describe how to consume a REST API service using Vue.JS
- describe how to perform API versioning using URIs
- describe schema first design using an API Specification Language
- describe the advantages and disadvantages of using either private or public APIs
- describe the advantages and disadvantages of using either XML or JSON for transferring and receiving data from Web Services
- describe the benefits and constraints when using web services as part of your business strategy
- describe the benefits of using a Cloud base backend servicer such as Firebase
- describe the best practices for implementing API resources
- describe the best practices when defining the API Schema and using verbs and nouns
- describe the best practices when defining URIs
- describe the relationship between URLs and URIs and when to use each
- describe the various standards for returning data from a web service
- describe the various use cases for using an API Gateway to deploy REST API web service
- describe URI templates and how they are used to provide guidelines for developers
- describe various methods for consuming web services from client-side devices
- describe what is JSON and how if can be used with REST APIs
- describe what is meant by Hypertext and Hypermedia and their differences
- identify URI templates and how they are used to provide guidelines for developers
- OAuth and API Security

API Developer

Objectives

- In this lab, you will perform API Developer tasks such as creating SOAP and REST web services, creating a JSON model for a Web API, and consuming a REST API service using JavaScript. Then, test your skills by answering assessment questions after exploring how to consume a REST API service using JQuery, demonstating HTML5 input types, using the Windows Workflow Foundation and authenicating an API using an external service.

Business & Leadership for API Developers

Optional

COURSE

Developing and Supporting an Agile Mindset

1062

COURSE

Encouraging Team Communication and...

1536

COURSE

The Essential Role of the Agile Product Owner

275

COURSE

Using Strategic Thinking to Consider the Big Picture

595

COURSE

Getting to the Root of a Problem

926

COURSE

Unleashing Personal and Team Creativity

842

COURSE

Contributing as a Virtual Team Member

2634

COURSE

Developing a Growth Mindset

2372

COURSE

Effective Team Communication

1220

Bookshelf

Optional

BOOK

API Development: A Practical Guide for Busines...

4

BOOK

Software Development, Design and Coding: With...

9

BOOK

REST API Development with Node.js: Manage and...

9

BOOK

Reusable Firmware Development: A Practical...

2

BOOK

Principles of Package Design: Creating Reusable Software...

BOOK

Oracle Visual Builder Cloud Service Revealed: Rapid...

3

BOOK

Inclusive Design Patterns: Coding Accessibility Into...

8

BOOK

Penetration Testing Essentials

17

BOOK

Coding for Penetration Testers: Building Better Too...

2

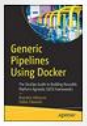
BOOK

Modern API Design with ASP.NET Core 2: Building...

BOOK

Pro RESTful APIs: Design, Build and Integrate with...

22



BOOK

Generic Pipelines Using Docker: The DevOps Guide...

2

BOOK

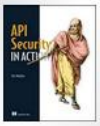
The Design of Web APIs

7

BOOK

Irresistible API's: Designing Web APIs That Developers...

2



BOOK

API Security in Action

1

Productivity Tools for API Developers

Optional

COURSE

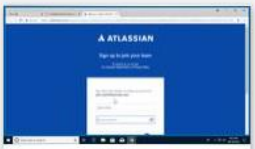
Planning & Working on a Software Project in Jira...

85

COURSE

Reporting in Jira Software

82



COURSE

Signing in & Navigating within Spaces

40

COURSE

Setting Up & Managing Spaces

33

COURSE

Working with Space

27

COURSE

Working with Team Members

75

COURSE

Configuring Spaces

20

Optional Resources

Optional

LAB

Front-End Developer Sandbox

Front-End Development 2020

7

LAB

C Programming Sandbox

C 18

11