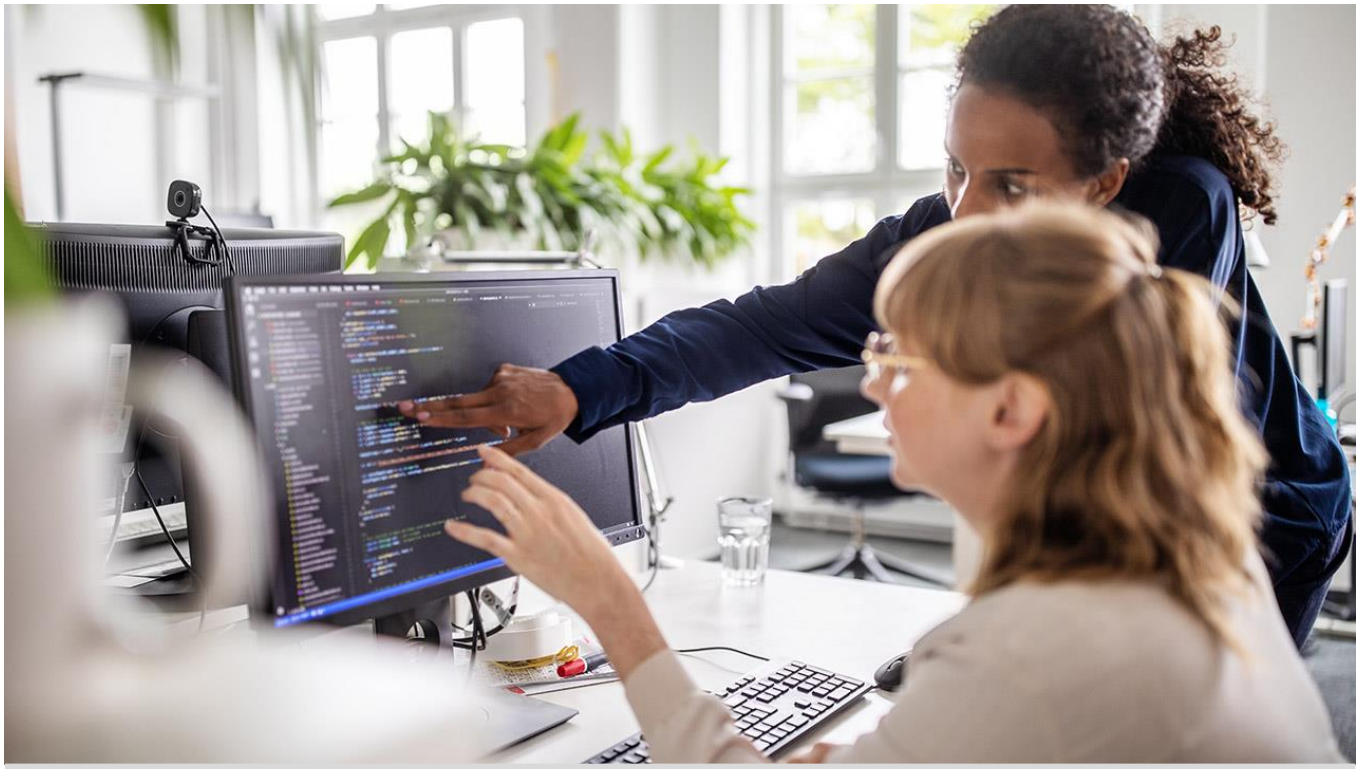


Web Programmer to Apprentice Programmer

SKILLSOFT ASPIRE JOURNEY

skillsoft | percipio

Tracks

Track 1: Web Programmer

In this Skillsoft Aspire track of the Web Programmer to Apprentice Programmer Journey, the focus will be on JavaScript.

Explore 10 courses | 12h 29m 43s 1 lab0

Track 2: Web Apps Developer

In this Skillsoft Aspire track of the Web Programmer to Apprentice Programmer Journey, the focus will be on building applications using React.

Explore 8 courses | 12h 32m 7s 1 lab0

Track 3: Novice Programmer

In this Skillsoft Aspire track of the Web Programmer to Apprentice Programmer Journey, the focus will be on SQL programming and building Web apps using Django.

Explore 15 courses | 18h 52m 29s 1 lab0

Track 4: Apprentice Programmer

In this Skillsoft Aspire track of the Web Programmer to Apprentice Programmer Journey, the focus will be on design patterns for JavaScript.

Explore 13 courses | 20h 26m 10s 1 lab0

PREREQUISITES

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

- Be familiar with programming
- Be familiar with Web applications
- Be familiar with databases

JavaScript: Introduction

Objectives:

- recognize where JavaScript fits within a larger web application involving HTML and CSS
- describe how browsers render web pages as a structured DOM
- list the high-level properties of the JavaScript programming language
- identify the properties and features around arrays and functions in JavaScript
- recall what an object is JavaScript and how it is represented
- identify some of the quirks of the JavaScript language which distinguish it from most other high-level languages
- recognize what closures are in JavaScript and what is included within this construct
- describe how common attributes of a set of objects can be defined with in a prototype for those objects

JavaScript: Getting Started with JavaScript Programming

Objectives:

- create a simple JavaScript app that writes a message to an HTML page
- recognize how JavaScript can be used to communicate with end users via the HTML page, alerts, and log messages
- link a HTML page with a separately defined JavaScript source file
- edit sections of a HTML page using JavaScript and initialize number and string variables
- identify the changes in behavior of a JavaScript app when Strict mode is enabled
- distinguish between the var, let, and const keywords when declaring variables
- differentiate variables that are undefined from those whose values are null
- use the alert, prompt, and confirm functions to relay information to and receive input from end users

JavaScript: Types, Operators, & Control Structures in JavaScript

Objectives:

- describe multiple ways to declare and manipulate numerical data in JavaScript
- initialize and manipulate JavaScript strings using different techniques
- use in-built string operations to analyze and modify string types
- create and manage date types in JavaScript
- use the constants, methods, and random number generators available in the JavaScript Math library
- apply the basic operators on numbers and strings
- perform logical tests using if statements and conditional operators
- convert data from one primitive type to another
- create while, do-while, and for loops in JavaScript

JavaScript: Functions & Arrays in JavaScript

Objectives:

- define basic functions and invoke them in JavaScript
- recognize how the var, let, and const keywords affect the accessibility of variables in JavaScript functions
- distinguish between the var and let keywords when it comes to the scope of JavaScript variables
- define functions using the concise syntax available in the ES6 specifications for JavaScript, including the use of arrow functions
- manage errors in your application, whether they are raised by JavaScript or by your own logic using try, catch, and finally blocks
- create arrays and access and iterate over their elements
- use the rest parameter syntax to create functions with an undefined number of parameters
- use the spread operator to expand the contents of arrays
- break down and access the components of arrays with array destructuring
- generate deep and shallow copies of arrays and add and remove array elements
- insert elements into the middle of arrays using splice and create slices of arrays using the slice method
- use the concat method to combine the contents of arrays and arrange array elements using the sort and reverse methods

JavaScript: Objects

Objectives:

- create objects containing different types of data and functions, and access their properties
- initialize an empty object and add/remove properties to/from that object
- recognize the role of the "this" keyword within the scope of a JavaScript object
- use the call method to link a JavaScript object to a function call
- implement an object constructor and use it to instantiate new objects
- use the assignment operator and the Object.assign() method to create new objects
- use the spread syntax and the JSON object to create new objects
- identify a number of object methods to access the keys and values of an object
- prevent changes from being made to an object by freezing it
- create a new array by performing an operation on the elements on an existing one using the map() method
- perform aggregate and filter operations on the contents of an array using the reduce() and filter() methods
- check the type of an object using the instanceOf operator

JavaScript: Advanced Function Operations

Objectives:

- assign a function to a variable and pass a function as an argument to another function
- define and invoke a function which returns a nested function
- recognize what data types in JavaScript are passed by value and by reference
- create a function from a string using the new keyword
- use function callbacks to ensure the sequential execution of functions
- use the call, apply, and bind methods to link a function invocation with an object

JavaScript: Closures & Prototypes

Objectives:

- recognize what makes up a closure in JavaScript
- recall how variables of the same name inside closures and in scopes outside them are accessed by functions
- distinguish between variables of the same name inside functions, within the closure, and ones defined in a global scope
- implement counter functions with closures
- integrate closures with buttons on a web page
- recognize how closures can be created within objects and its implications on your code
- recognize how closures created within loops interact with variables defined within and outside the loop
- implement getter and setter functions for JavaScript objects using closures
- identify when to use prototypes for JavaScript objects and how they can be defined
- distinguish between object prototypes and classes in JavaScript
- use prototypes to implement inheritance in JavaScript
- implement multiple levels of inheritance using prototypes

JavaScript: Working with the DOM & Events

Objectives:

- use the document object to retrieve individual elements of your web page and access their values and properties
- alter the contents of a web page by modifying its DOM
- retrieve the children of elements of your web page as an array
- insert new elements into your web page by adding nodes to the DOM structure
- delete or replace existing nodes in the DOM of your web page
- write a function to generate an animation by periodically updating the DOM of your web page
- define event listeners and handlers for clicks on a web page button
- create handlers for the events of users clicking on a button and for modifying the contents of a text box
- define handlers for mouse events on elements of your web page
- use the screen, window, and navigator objects of the BOM to get information about the end user's environment
- use the window object to interact with the end user's browser

JavaScript: Form Validation & Async Programming

Objectives:

- define a simple registration form with text, numerical input fields, radio buttons, check boxes and a drop-down menu
- access individual form elements and their data contents from JavaScript
- recognize when variables are destroyed by JavaScript when a form submission navigates the user to a different web page
- define a variety of form validations for different input fields
- identify operations which run asynchronously in your JavaScript application
- use promises to chain the execution of two functions where one depends on the results returned by the other
- define a function making multiple asynchronous function calls using the async/await syntax
- use the fetch function to asynchronously retrieve an image file and relate it to the XMLHttpRequest of AJAX

Final Exam: Web Programmer

Objectives:

- access individual form elements and their data contents from JavaScript
- alter the contents of a web page by modifying its DOM
- apply basic operators on numbers and strings
- assign a function to a variable and pass a function as an argument to another function
- create a function from a string using the new keyword
- create and manage date types in JavaScript
- create a simple JavaScript app that writes a message to an HTML page
- create handlers for the events of users clicking on a button and for modifying the contents of a text box
- create objects containing different types of data and functions, and access their properties
- define a simple registration form with text, numerical input fields, radio buttons, check boxes and a drop-down menu
- define a variety of form validations for different input fields
- define basic functions and invoke them in JavaScript
- define event listeners and handlers for clicks on a web page button
- define functions using the concise syntax available in the ES6 specifications for JavaScript, including the use of arrow functions
- define handlers for mouse events on elements of your web page
- delete or replace existing nodes in the DOM of your web page
- describe how browsers render web pages as a structured DOM
- describe how browsers render web pages as a structured DOM
- describe multiple ways to declare and manipulate numerical data in JavaScript
- differentiate variables that are undefined from those whose values are null
- distinguish between the var and let keywords when it comes to the scope of JavaScript variables
- distinguish between the var, let, and const keywords when declaring variables
- generate deep and shallow copies of arrays and add and remove array elements
- identify a number of object methods to access the keys and values of an object

- identify some of the quirks of the JavaScript language which distinguish it from most other high-level languages
- identify the characteristics of an array
- identify the properties and features around arrays and functions in JavaScript
- identify when to use prototypes for JavaScript objects and how they can be defined
- implement an object constructor and use it to instantiate new objects
- implement basic operators on numbers
- implement counter functions with closures
- initialize and manipulate JavaScript strings using different techniques
- initialize an empty object and add/remove properties to/from that object
- integrate closures with buttons on a web page
- link a HTML page with a separately defined JavaScript source file
- list the high-level properties of the JavaScript programming language
- manage errors in your application, whether they are raised by JavaScript or by your own logic using try, catch, and finally blocks
- recall how variables of the same name inside closures and in scopes outside them are accessed by functions
- recall what an object is JavaScript and how it is represented
- recognize handlers for mouse events on elements of your web page
- recognize how closures can be created within objects and its implications on your code
- recognize how closures created within loops interact with variables defined within and outside the loop
- recognize how JavaScript can be used to communicate with end users via the HTML page, alerts, and log messages
- recognize how the var, let, and const keywords affect the accessibility of variables in JavaScript functions
- recognize the role of the "this" keyword within the scope of a JavaScript object
- recognize what data types in JavaScript are passed by value and by reference
- recognize what makes up a closure in JavaScript
- recognize when variables are destroyed by JavaScript when a form submission navigates the user to a different web page
- recognize where JavaScript fits within a larger web application involving HTML and CSS
- retrieve the children of elements of your web page as an array
- use function callbacks to ensure the sequential execution of functions
- use in-built string operations to analyze and modify string types
- use promises to chain the execution of two functions where one depends on the results returned by the other
- use prototypes to implement inheritance in JavaScript
- use the assignment operator and the Object.assign() method to create new objects
- use the call method to link a JavaScript object to a function call
- use the document object to retrieve individual elements of your web page and access their values and properties
- use the fetch function to asynchronously retrieve an image file and relate it to the XMLHttpRequest of AJAX
- use the spread operator to expand the contents of arrays
- use the spread syntax and the JSON object to create new objects

Web Programmer

Objectives:

- In this practice lab, learners will be presented with a series of exercises to practice web programming with JavaScript. Exercises include tasks such as setting up an effective JavaScript Environment, and working with JS types and arrays. Learners will also practice implementing call back functions, adding closure and callback functions, adding a form to capture user values, and achieving asynchronous behavior with promises.

Build Apps Using React: Introducing React for Web Applications

Objectives:

- describe what the React library is used for
- design components using composition
- identify the features that make React powerful
- describe the concepts underlying important React features
- describe what the virtual DOM is and know how it is used
- implement a simple web page using React APIs
- recognize the basic components of the web page built using React
- use minified, productionized React libraries
- create a DOM hierarchy with nested components
- use the key attribute to identify unique sibling elements
- describe what JSX is and its important features
- describe the functionality of the Babel compiler when used with JSX
- create elements using JSX and the Babel compiler
- render a DOM hierarchy using JSX
- evaluate expressions within JSX
- refer to JavaScript variables within JSX

Build Apps Using React: Local Machine & Online Playgrounds

Objectives:

- install the NodeJS libraries and the npm package manager on MacOS
- install the NodeJS libraries and the npm package manager on Windows
- run http-server on the local machine to serve web pages
- use ReactDOM.render() to render elements
- recognize smart rendering of elements in React
- build simple components using ES6 classes
- build multiple components using ES6 classes
- implement a simple application on CodePen
- configure the Babel preprocessor on CodePen
- build a simple React prototype on CodePen
- prototype a simple React application on CodeSandbox
- prototype a simple React application on Glitch
- install and use create-react-app utilities on the local machine
- create components using functions
- compose multiple simple components to create a complex component

Build Apps Using React: Props & State

Objectives:

- identify props and state in components and describe when they are used
- recall the differences between props and state
- pass data into components using props
- evaluate expressions to pass data to components
- transfer props from higher to lower level components manually
- transfer props from higher to lower level components using the spread operator
- dynamically render components based on conditions
- specify default values passed in as props
- perform specific validations on props
- access child elements using props.children
- evaluate expressions to pass values to props
- specify functions as children of a component
- store data within a component using state
- update state using this.setState() in a component
- wire up event handlers to update the state of a component
- access the previous state of a component

Build Apps Using React: Working with Events

Objectives:

- recall the use of event propagation in the capture and bubble phases
- describe the advantages of synthetic events
- recognize the phases in a React component's lifecycle
- describe how DOM reconciliation works
- wire up event handlers to elements
- prevent the browser's default event handling
- use synthetic events to access event properties
- build a component to toggle state
- summarize what happens in the mounting phase of a component's lifecycle
- execute code once a component is mounted to the DOM
- summarize what happens in the unmounting phase of a component's lifecycle
- hook into the update phase of a component's lifecycle
- optimize the rendering of a component
- apply error boundaries to catch errors

Build Apps Using React: Working with Forms

Objectives:

- recall the use of event propagation in the capture and bubble phases
- describe the advantages of synthetic events
- recognize the phases in a React component's lifecycle
- describe how DOM reconciliation works
- wire up event handlers to elements
- prevent the browser's default event handling
- use synthetic events to access event properties
- build a component to toggle state
- summarize what happens in the mounting phase of a component's lifecycle
- execute code once a component is mounted to the DOM
- summarize what happens in the unmounting phase of a component's lifecycle
- hook into the update phase of a component's lifecycle
- optimize the rendering of a component
- apply error boundaries to catch errors

Build Apps Using React: Advanced Features in React

Objectives:

- recall how controlled components work in forms
- wire up event handlers for form mutations
- synchronize component state using event handlers
- wire up a single event handler for multiple inputs
- design form elements to be individual components
- test the form on a web browser
- recall the need for client-side validation
- validate data passed into forms
- compare controlled and uncontrolled components for form data
- use uncontrolled components to handle file inputs in forms

Build Apps Using React: In Development & Production

Objectives:

- describe the error that occurs when list elements are rendered without keys
- specify unique keys for list elements
- sketch the right usage of keys in components
- render components using conditional evaluation
- perform conditional rendering using variables
- evaluate conditions using inline logical operators
- perform rendering using ternary operators
- illustrate the use of local state in components
- recall the disadvantages of storing state locally
- extract state from lower level components
- store state in higher level components for synchronization
- reuse code using inheritance
- reuse code using composition
- customize child elements using composition
- create special components using composition
- illustrate the use of global properties without context
- modify components to use context for global properties

Final Exam: Web Apps Developer

Objectives:

- access the previous state of a component
- apply error boundaries to catch errors
- build simple components using ES6 classes
- compare controlled and uncontrolled components for form data
- configure the Babel preprocessor on CodePen
- create a production build with minified files
- describe how DOM reconciliation works
- describe the advantages of synthetic events
- describe the concepts underlying important React features
- describe the functionality of the Babel compiler when used with JSX
- describe what JSX is and its important features
- describe what the React library is used for
- describe what the virtual DOM is and know how it is used
- design components using composition
- develop a simple application in the create-react-app environment
- evaluate conditions using inline logical operators

- execute code once a component is mounted to the DOM
- extract state from lower-level components
- identify props and state in components and describe when they are used
- identify the features that make React powerfully
- illustrate the use of local state in components
- implement a simple application on CodePen
- implement a simple web page using React APIs
- install and use create-react-app utilities on the local machine
- install and work with React Developer Tools
- install the NodeJS libraries and the npm package manager on MacOS
- install the NodeJS libraries and the npm package manager on Windows
- pass data into components using props
- perform conditional rendering using variables
- perform specific validations on props
- prevent the browser's default event handling
- recall how controlled components work in forms
- recall the differences between props and state
- recall the need for client-side validation
- recall the use of event propagation in the capture and bubble phases
- recognize smart rendering of elements in React
- recognize the basic components of the web page built using React
- recognize the phases in a React component's lifecycle
- render components using conditional evaluation
- reuse code using composition
- reuse code using inheritance
- run http-server on the local machine to serve web pages
- sketch the right usage of keys in components
- specify default values passed in as props
- specify unique keys for list elements
- store data within a component using state
- store state in higher-level components for synchronization
- summarize what happens in the mounting phase of a component's lifecycle
- test the form on a web browser
- transfer props from higher to lower-level components manually
- update state using this.setState() in a component
- use minified, productized React libraries
- use ReactDOM.render() to render elements
- use state within the application
- validate data passed into forms
- wire up a single event handler for multiple inputs
- wire up event handlers to add new comments
- wire up event handlers to delete comments
- wire up event handlers to elements
- wire up event handlers to update the state of a component

Web Apps Developer

Objectives:

- In this practice lab, learners will be presented with a series of exercises to practice web app development using React. Exercises include tasks such as installing ReactJS libraries and rendering components, building custom components, adding state properties to a class component, and taking advantage of a component's life cycle methods. Learners will also practice working with forms, using appropriate keys when handling lists, installing an app from basic files, and handling the rendering of JSX.

MySQL: Getting Started

Objectives:

- identify requirements that a database management system needs to satisfy
- install MySQL on different platforms
- connect to a MySQL server using a command line interface
- create a database and view a list of databases on a MySQL server
- install MySQL workbench and connect to a MySQL server
- model entities in a real-world scenario for use in a database system
- execute commands from the MySQL Workbench environment
- perform basic database operations such as inserting and querying data
- view table metadata and delete data from tables

MySQL: Creating & Updating Tables

Objectives:

- enumerate attributes of keys, super-keys, and candidate keys
- model relationships in a real-world scenario for use in a database system
- specify non-null constraints while creating tables
- run simple queries using wildcards and where clauses
- understand the need for SQL and enumerate its important characteristics
- designate columns as unique while creating tables
- perform update operations on data in a MySQL relation
- perform DDL operations including altering tables to add primary key constraints, and dropping tables and databases

MySQL: Querying Data

Objectives:

- translate entity-relationship models to actual database table schemas
- identify the correct data types and constraints for specific columns in a table schema
- use relational and logical operators in the WHERE clause of MySQL queries
- use the LIKE and IN operators as well as wildcards in queries
- use subqueries to perform complex logical operations
- implement referential integrity checks using foreign key constraints
- perform filtering operations on date columns
- use the LIMIT and ORDER BY clauses in MySQL queries

MySQL: Understanding & Implementing Joins

Objectives:

- identify the use cases of joins
- compute the cross-join of two relations
- compute the inner join of two relations
- compute the left outer join of two relations
- compute the right outer join of two relations
- define the natural join operator
- implement a cross product in MySQL
- perform join operations in MySQL

MySQL: Grouping & Aggregation Operations

Objectives:

- prepare a small but meaningful relation that can be used to work with data aggregation operators
- use the DISTINCT keyword as well as the GROUP BY clause to identify duplicates and cardinality
- use the GROUP BY clause with important common aggregation functions such as SUM, COUNT, MIN, MAX
- use the HAVING keyword along with the GROUP BY clause to apply filters to aggregates
- use the SOME, ALL, and ANY keywords to link subqueries to outer queries
- implement joins with more than two relations in MySQL

MySQL: Views, Indices, & Normal Forms

Objectives:

- define a view and enumerate applications of views
- implement views in MySQL
- use views to abstract complex queries in MySQL
- define a database index and enumerate advantages and disadvantages of indexing columns in a relation
- view indices on an existing table and create new indices on columns
- create composite indices and drop indices
- identify and apply the first three normal forms of database design
- evaluate the number of tables needed to appropriately store and model data
- identify and rectify violations of the First Normal Form in MySQL
- identify violations of the Second Normal Form in MySQL
- rectify violations of the Second Normal Form in MySQL by refactoring table design

MySQL: Triggers & Stored Procedures

Objectives:

- define a trigger and enumerate use cases of triggers in a database system
- create a row-level trigger executed after inserts into a MySQL table
- create a row-level trigger executed before inserts into a MySQL table
- use triggers to performing cascading inserts and satisfy foreign key constraints
- override default ON DELETE and ON UPDATE behavior in MySQL tables
- enumerate applications of stored procedures
- create a simple stored procedure correctly specifying choice of delimiter
- invoke a stored procedure and pass in required arguments
- check if a stored procedure already exists and drop it before re-defining it
- combine complex declarative and imperative operations in a stored procedure

MySQL: Transactions, Savepoints, & Locks

Objectives:

- enumerate the ACID properties and their importance
- start a transaction and execute statements within it
- rollback insert and update commands executed within a transaction
- identify commands that can not be rolled back once executed
- enumerate DDL operations which are immediately reflected to database state
- leverage stored procedures to control transaction execution
- specify custom handlers for errors and warnings occurring within transactions
- use savepoints to perform rollback to specific points within a transaction
- understand precise semantics of sequential savepoints in transactions
- create multiple client sessions connected to the same MySQL server
- acquire a read lock on a relation and understand the semantics of read locking
- acquire a write lock on a relation and understand the semantics of write locking

Building Web Apps Using Django: Introduction to Web Frameworks & Django

Objectives:

- recall the steps and software required when processing web requests for static and dynamic websites
- list the tasks involved in building a website and how web frameworks can speed up the process
- recognize the Django framework features that can help to simplify web development
- identify the components of a Django application that are involved in processing web requests
- describe what templates are in the context of Django and list their use cases
- compare Django models to database tables and describe the role of the Django ORM in mapping the two
- enumerate some of the built-in Django apps that developers can integrate into their own projects

Building Web Apps Using Django: Building a Basic Website

Objectives:

- create a virtual Python environment and install Django in it
- generate a new Django project and describe the various files that are created by this operation
- start the built-in Django development server on the default port, as well a specified port
- define a view and a URL pattern in Django to render the text "Hello World" in a web page
- generate a new app within a Django project
- use the manage.py script of a Django project to propagate model definitions to the database
- configure project and app-level URLs in Django
- define a view that renders an HTML file in its response
- download and use boilerplate HTML, CSS, and JavaScript files in a Django project
- modify boilerplate HTML files to suit the requirements of a Django project

Building Web Apps Using Django: Templates & User Administration

Objectives:

- define the URL, view, and template for a new web page in a Django application
- use the Django template language to generate a URL for a URL pattern
- create a base template containing the common elements for multiple pages in your Django web site
- extend the base template and the common elements defined within it while building individual pages in your Django web site
- convey data from a view to a Django template in a dictionary
- apply styles defined in your CSS files to Django templates
- create a superuser for your Django project and sign in to the Django user administration app
- use the Django Admin interface to create new users and groups
- describe the effects of assigning staff user status and specific permissions to Django app users

Building Web Apps Using Django: Models & User Registration

Objectives:

- define an entity in your Django application in the form of a model
- propagate a model definition to your project's database by generating and running a migration script
- create instances of a Django model and access model attributes, including related objects
- view, update, and create instances of your Django model from the built-in admin interface
- create a view that uses Django's built-in user registration form
- define the URL and template file for the user registration page for a Django application
- convey notifications to your website's users with flash messages
- save the data submitted in a user registration form to the Django project database
- add additional fields in the user registration form by extending its definition
- install and use the django-crispy-forms library to format your Django application's user registration page

Building Web Apps Using Django: Implementing Login & Logout

Objectives:

- build a user login page in a website using built-in Django objects
- configure where a user is redirected to on a Django website upon a successful login
- define a logout template to serve as the logout page for your Django website
- present different views for users who are signed in to your Django website as opposed to regular users
- configure views in your Django application to render only when the user is signed in
- define the model for the profile of a user of your Django application
- set the location in your Django project directory where media can be stored
- display a user's information, including their associated image, in a profile page

Building Web Apps Using Django: Generic Views

Objectives:

- define forms and a view function to enable users of your Django web site to update model instances
- configure update forms so that users can submit the new data for model instances in a POST request
- use the built-in generic ListView to list instances of a model in your Django project
- display all details of a Django model instance using the built-in DetailView view
- enable the creation of new instances of your Django model by implementing a CreateView view
- allow users to update instances of your Django model by means of an UpdateView view
- use the built-in DeleteView view to allow users to delete instances of your Django model
- set permissions for users to delete instances of your Django model and define the redirect URL for a successful delete
- generate an archive of your Django model instances using the ArchiveIndexView view
- create an archive of your Django model instances, categorized by year, using the YearArchiveView view
- define an "About Us" page for your Django application
- set a customized 404 error page for your Django web site

Final Exam: Novice Programmer

Objectives:

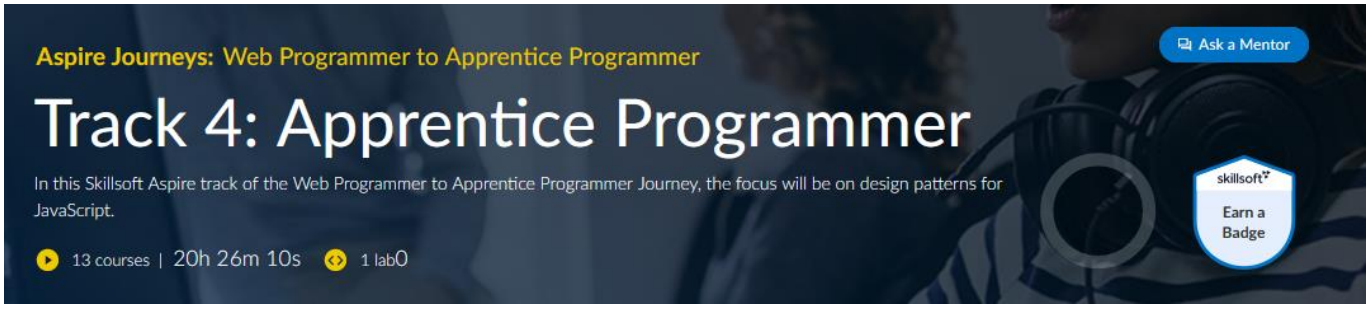
- allow users to update instances of your Django model by means of a UpdateView view
- build a user login page in a web site using built-in Django objects
- compute the cross-join of two relations
- compute the inner join of two relations
- compute the left outer join of two relations
- connect to a MySQL server using a command line interface
- convey notifications to your web site's users with flash messages
- create a row-level trigger executed after inserts into a MySQL table
- create a row-level trigger executed before inserts into a MySQL table
- create a superuser for your Django project and sign in to the Django user administration app
- create a virtual Python environment and install Django in it
- create composite indices and drop indices
- define a database index and enumerate advantages and disadvantages of indexing columns in a relation
- define a logout template to serve as the logout page for your Django web site
- define an entity in your Django application in the form of a model
- define a trigger and enumerate use cases of triggers in a database system
- define a view and enumerate applications of views
- define the model for the profile of a user of your Django application
- define the natural join operator
- define the URL, view, and template for a new web page in a Django application
- display all details of a Django model instance using the built-in DetailView view
- enable the creation of new instances of your Django model by implementing a CreateView view
- enumerate applications of stored procedures
- enumerate attributes of keys, super-keys, and candidate keys
- enumerate the ACID properties and their importance
- generate a new app within a Django project
- generate a new Django project and describe the various files that are created by this operation
- identify commands that cannot be rolled back once executed
- identify requirements that a database management system needs to satisfy
- identify the correct data types and constraints for specific columns in a table schema
- implement referential integrity checks using foreign key constraints
- implement views in MySQL
- install MySQL on different platforms
- invoke a stored procedure and pass in required arguments
- leverage stored procedures to control transaction execution
- list the tasks involved in building a web site and how web frameworks can speed up the process
- model relationships in a real-world scenario for use in a database system
- prepare a small but meaningful relationships that can be used to work with data aggregation operators
- present different views for users who are signed in to your Django web site as opposed to regular users

- propagate a model definition to your project's database by generating and running a migration script
- recall the steps and software required when processing web requests for static and dynamic web sites
- recognize MySQL default behaviour
- recognize the Django framework features that can help to simplify web development
- rollback insert and update commands executed within a transaction
- run simple queries using wildcards and where clauses
- save the data submitted in a user registration form to the Django project database
- set permissions for users to delete instances of your Django model and define the redirect URL for a successful delete
- specify non-null constraints while creating tables
- start a transaction and execute statements within it
- start the built-in Django development server on the default port, as well a specified port
- use relational and logical operators in the WHERE clause of MySQL queries
- use the built-in generic ListView to list instances of a model in your Django project
- use the DISTINCT keyword as well as the GROUP BY clause to identify duplicates and cardinality
- use the Django Admin interface to create new users and groups
- use the Django template language to generate a URL for a URL pattern
- use the GROUP BY clause with important common aggregation functions such as SUM, COUNT, MIN, MAX
- use the LIKE and IN operators as well as wildcards in queries
- use the manage.py script of a Django project to propagate model definitions to the database
- use views to complex abstract queries in MySQL
- view, update and create instances of your Django model from the built-in admin interface

Novice Programmer

Objectives:

- In this practice lab, learners will be presented with a series of exercises to practice SQL programming. Exercises include tasks such as running MySQL Workbench, adding, and removing a constraint to a table, creating a unique field, and writing queries. Learners will also practice using built-in functions, working with indexes, preparing a trigger, and writing a stored procedure.



Design Patterns in JavaScript: Getting Started

Objectives:

- define design patterns
- describe the principles of good design
- summarize principles of good design and architecture
- recall anti-patterns and their uses
- recall programming anti-patterns
- recall JavaScript-specific anti-patterns
- compare the different types of design patterns
- install required tools on a MacOS machine
- install required tools on a Windows machine
- recall key concepts covered in this course

Design Patterns in JavaScript: Constructor, Factory, & Abstract Factory Creational Patterns

Objectives:

- describe the key features of the constructor pattern
- use the object literal notation to construct objects
- assign properties and functions to objects
- use Object.create() and new Object() to create objects
- create objects using functions
- create objects using ES6 classes
- describe the Factory and abstract factory patterns
- define helper functions for the factory pattern
- setting up helper methods to implement the Factory pattern
- implement the Factory pattern
- set up constructs and factories for the abstract factory pattern
- define immediately-invoked functions for the factory
- implement the abstract factory pattern

Design Patterns in JavaScript: Singleton, Prototype, & Builder Creational Patterns

Objectives:

- recall the basic principles of the Singleton pattern
- define helper functions for the Singleton pattern
- implement the Singleton pattern
- describe features of the Prototype pattern
- implement the Prototype pattern
- apply best practices for the Prototype pattern
- implement the Prototype pattern without using Object.create()
- recall characteristics of the Builder pattern
- implement the Builder pattern
- use the Builder pattern to construct complex objects
- implement the Builder pattern in jQuery
- use different methods in the iQuery Builder pattern
- recall the key concepts covered in this course

Design Patterns in JavaScript: Module, Revealing Module, Façade, Decorator, & Mixin Structural Patterns

Objectives:

- describe the Module and Revealing Module patterns
- implement private variables with the Module pattern
- implement private functions with the Module pattern
- use the Revealing Module pattern
- apply best practices for working with the Revealing Module pattern
- describe the Façade pattern
- set up helpers for the Façade pattern
- implement the Façade pattern
- make AJAX requests without a façade in jQuery
- use the Façade pattern in jQuery for AJAX requests
- recall the principles of the Decorator pattern
- set up helpers for the Decorator pattern
- apply the decorators for dynamic customizations
- design React components as decorators
- use React components as decorators
- describe the Mixin pattern
- use Mixins in the Underscore.js library
- simulate multiple inheritance with Mixins
- implement the Mixin pattern
- recall the key concepts covered in this course

Design Patterns in JavaScript: Flyweight, Adapter, Composite, & Proxy Structural Patterns

Objectives:

- recall the characteristics of the Flyweight pattern
- describe how event handling works on the browser
- illustrate inefficient memory usage for granular objects
- share resources using the Flyweight pattern
- wire up individual event handlers to HTML elements
- centralize event handling using the Flyweight pattern
- articulate features of the Adapter pattern
- illustrate the burden on the client when the Adapter pattern is not used
- provide a consistent client interface using an adapter
- recall the characteristics of the Composite pattern
- apply the Composite pattern using jQuery
- describe the Proxy pattern
- use the Proxy pattern to cache data on the client
- execute functions using the right context
- apply the Proxy pattern to provide the right context

Design Patterns in JavaScript: Observer & Iterator Behavioral Patterns

Objectives:

- recall the key characteristics of the Observer pattern
- implement publishers and subscribers
- publish messages using the publisher and receive them at subscribers
- recognize the role of the Observer pattern in event handling
- trigger custom events in jQuery
- design the Observer with custom events in jQuery
- describe characteristics of the Iterator pattern
- implement the Iterator mixin
- iterate over elements in a collection
- describe Iterators in jQuery
- recall the key concepts covered in this course

Design Patterns in JavaScript: Mediator, State, & Command Behavioral Patterns

Objectives:

- recall the characteristics of the Mediator pattern
- set up the Mediator as a workflow object
- use the Mediator for communication
- describe the characteristics of the State pattern
- set up helper methods for the State pattern
- implement the State pattern in React
- perform valid state transitions
- describe the characteristics of the Command pattern
- implement the Command pattern
- execute and undo commands

Unit Testing in JavaScript: Mocha & Unit.js

Objectives:

- install and configure the Mocha testing environment
- install the Unit.js assertion library
- use assertions to test boolean values
- use assertions to test strings
- match strings using regular expressions and custom functions
- compare numeric values in tests
- use assertions to test function objects
- work with arrays and array elements
- compare objects using equality operations
- test objects using match
- determine whether objects are instances of other objects
- summarize the different ways to test errors
- describe how values of different data types can be tested

Unit Testing in JavaScript: Should.js & Must.js

Objectives:

- recall the different styles that can be used with the Should.js API
- describe how different data types can be tested using Should.js
- execute assertions to check for values that evaluate to true and false using Should.js
- perform pattern matching and substring checks using Should.js
- evaluate numeric values using Should.js
- examine objects and properties using Should.js
- evaluate functions that return promises
- recall the different styles that can be used with the Must.js API
- run basic assertions using Must.js
- use assertions to test different data types with Must.js

Unit Testing in JavaScript: Exploring & Configuring the Mocha Testing Framework

Objectives:

- recognize the use of the before() and after() hooks in tests
- recognize the use of the beforeEach() and afterEach() hooks in tests
- implement tests using multiple hooks
- recall why the use of the arrow functions should be avoided in Mocha
- use the done() method to signal test completion
- set up asynchronous test functions
- perform asynchronous testing using async/await
- prepare Mocha tests to run within a browser
- run Mocha tests in a browser
- configure different reporters in Mocha
- run multiple tests from the command line

Unit Testing in JavaScript: Mocha & Chai

Objectives:

- install the Chai library on your local machine
- use assertions to test boolean values
- compare numeric values using assert
- performing string comparisons using assert
- describe the assertions that can be used to test objects and their properties
- use arrays and elements in arrays using assert
- compare and contrast the should and expect APIs in Chai
- test different data types using should
- test different data types using expect
- perform asynchronous function testing using expect

Unit Testing in JavaScript: SinonJS

Objectives:

- install Mocha and Sinon on your local machine
- create an object to be used in test cases
- use fakes to record function invocations
- configure fakes with specific behavior
- test the order of function invocations using fakes
- recognize how to test errors and callbacks with fakes
- perform asynchronous tests using fakes with promises
- create and use anonymous spies for testing
- wrap spies around existing functions
- spy on specific methods of an object
- describe the use of sandboxes for testing
- perform testing using spies on object getters and setters
- test real world components using spies
- configure stub behavior in test cases
- create stubs for object methods
- instantiate objects without invoking object constructors
- perform testing using mocks and set expectations
- use spy calls to test function invocations
- control time in tests to make asynchronous testing easy
- use fake timers with multiple async functions
- use sandboxes to simplify testing

Final Exam: Apprentice Programmer

Objectives:

- articulate features of the Adapter pattern
- compare and contrast the should and expect APIs in Chai
- compare numeric values in tests
- compare numeric values using Assert
- configure different reporters in Mocha
- configure fakes with specific behavior
- create an object to be used in test cases
- create stubs for object methods
- define design patterns
- define helper functions for the factory pattern
- describe features of the Prototype pattern

- describe how different data types can be tested using Should.js
- describe how event handling works on the browser
- describe the characteristics of the Command pattern
- describe the characteristics of the Iterator pattern
- describe the characteristics of the State pattern
- describe the Façade pattern
- describe the Factory and abstract factory patterns
- describe the key features of the constructor pattern
- describe the Mixin pattern
- describe the Module and Revealing Module patterns
- describe the principles of good design
- describe the Proxy pattern
- describe the use of sandboxes for testing
- determine whether objects are instances of other objects
- identify principles of good design
- illustrate inefficient memory usage for granular objects
- implement private functions with the Module pattern
- implement private variables with the Module pattern
- implement publishers and subscribers
- implement the Builder pattern
- implement the Iterator mixin
- implement the Prototype pattern
- install and configure the Mocha testing environment
- install Mocha and Sinon on your local machine
- install the Chai library on your local machine
- install the Unit.js assertion library
- perform asynchronous tests using fakes with promises
- perform pattern matching and substring checks using Should.js
- prepare Mocha tests to run within a browser
- recall characteristics of the Builder pattern
- recall the basic principles of the Singleton pattern
- recall the characteristics of the Composite pattern
- recall the characteristics of the Flyweight pattern
- recall the characteristics of the Mediator pattern
- recall the different styles that can be used with the Should.js API
- recall the key characteristics of the Observer pattern
- recall the principles of the Decorator pattern
- recognize the use of the beforeEach() and afterEach() hooks in tests
- run basic assertions using Must.js
- run Mocha tests in a browser
- set up helpers for the Decorator pattern
- set up the Mediator as a workflow object
- use assertions to test boolean values
- use fakes to record function invocations
- use Mixins in the Underscore.js library
- use Object.create() and new Object() to create objects
- use sandboxes to simplify testing
- use the literal object notation to construct objects
- work with arrays and array elements

Apprentice Programmer

Objectives:

- In this practice lab, learners will be presented with a series of exercises to practice working with design patterns for JavaScript. Exercises include tasks such as building JavaScript objects, constructing parameterized objects, converting a functional object, and using a Factory method. Learners will also practice creating a Singleton object, implementing the Facade pattern, using the Flyweight pattern, and subscribing using the Observer pattern.

Productivity Tools for Apprentice Programmers

Optional

COURSE

Signing in & Setting up a Team

21

COURSE

Using the Conversation Tools

20

COURSE

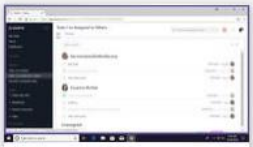
Creating & Managing Projects

17

COURSE

Finding & Sharing Items

10



COURSE

Running Reports & Configuring Projects

11

COURSE

Signing In & Setting Up

28

COURSE

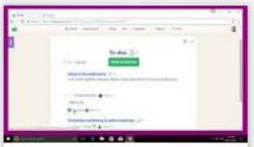
Using the Team Communication Tools

82

COURSE

Setting Up & Tracking Projects

25



COURSE

Managing your Project Tasks & Assets

20

COURSE

Using the Calendar Tools

22

COURSE

Using Basecamp for iOS

21

COURSE

Sign-in & Setup

18

COURSE

Communication Tools

23

COURSE

Working with Groups

14

COURSE

Creating, Finding, & Sharing Information

12

COURSE

Configuring Convo

6

COURSE

The Convo iOS App

13

COURSE

Sign-in & Setup

65

COURSE

Creating Teams & Boards

43

COURSE

Managing Cards

22

COURSE

Finding & Sharing Information

20

COURSE

Setting Up

61

COURSE

Posting & Reacting to Status Updates

46

COURSE

Using Groups

14

COURSE

Collaborating & Communicating

82

Productivity Tools for Apprentice Programmers Continued

COURSE

Configuring Networks

25

COURSE

Creating & Setting Up Projects in Jira Cloud

164

COURSE

Configuring & Managing Boards in Jira Cloud

106

COURSE

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

85

COURSE

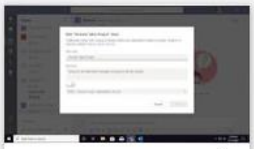
Reporting in Jira Software

83

COURSE

Getting to know the application

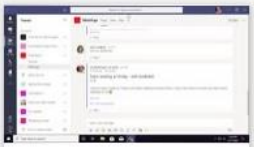
1010



COURSE

Using Teams & Channels

746



COURSE

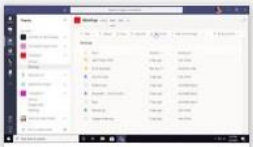
Communicating via the App

730

COURSE

Formatting, Illustrating & Reacting to Messages

543



COURSE

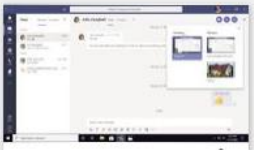
Creating, Finding & Organizing Files

536

COURSE

Working with Apps, Tabs & Wiki

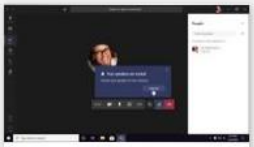
465



COURSE

Making calls, Organizing Contacts & Using Voicemail

453



COURSE

Creating, Joining & Managing Meetings

462

Business & Leadership for Apprentice Programmers

Optional

COURSE

Developing and Supporting an Agile Mindset

1064

COURSE

Encouraging Team Communication and...

1542

COURSE

The Essential Role of the Agile Product Owner

276

COURSE

Using Strategic Thinking to Consider the Big Picture

599

COURSE

Getting to the Root of a Problem

926

COURSE

Unleashing Personal and Team Creativity

844

COURSE

Contributing as a Virtual Team Member

2636

COURSE

Developing a Growth Mindset

2382

COURSE

Effective Team Communication

1225

COURSE

Reaching Sound Conclusions

275

Bookshelf

Optional

BOOK

Beginning JavaScript: The Ultimate Guide to Modern...

32

BOOK

Full Stack JavaScript: Learn Backbone.js, Node.js, and...

13

BOOK

Front-End Reactive Architectures: Explore the...

5

BOOK

Pro JavaScript Techniques, Second Edition

7

BOOK

Professional JavaScript for Web Developers, 4th Edition

8

BOOK

Functional Programming in JavaScript

4

BOOK

Get Programming with JavaScript

3

BOOK

Get Programming with JavaScript Next: New...

2

BOOK

JavaScript: A Beginner's Guide, Fifth Edition

16

BOOK

Eloquent JavaScript: A Modern Introduction to...

24

BOOK

Secrets of the JavaScript Ninja, Second Edition

5

BOOK

React Native in Action: Developing iOS and Androi...

10

BOOK

Introduction to React

15

BOOK

Pro MERN Stack: Full Stack Web App Development wit...

12

BOOK

Pro React 16

23

BOOK

React Native for iOS Development

8

BOOK

React Native for Mobile Development: Harness the...

17

BOOK

React in Action

13

BOOK

React Quickly: Painless Web Apps with React, JSX, Redu...

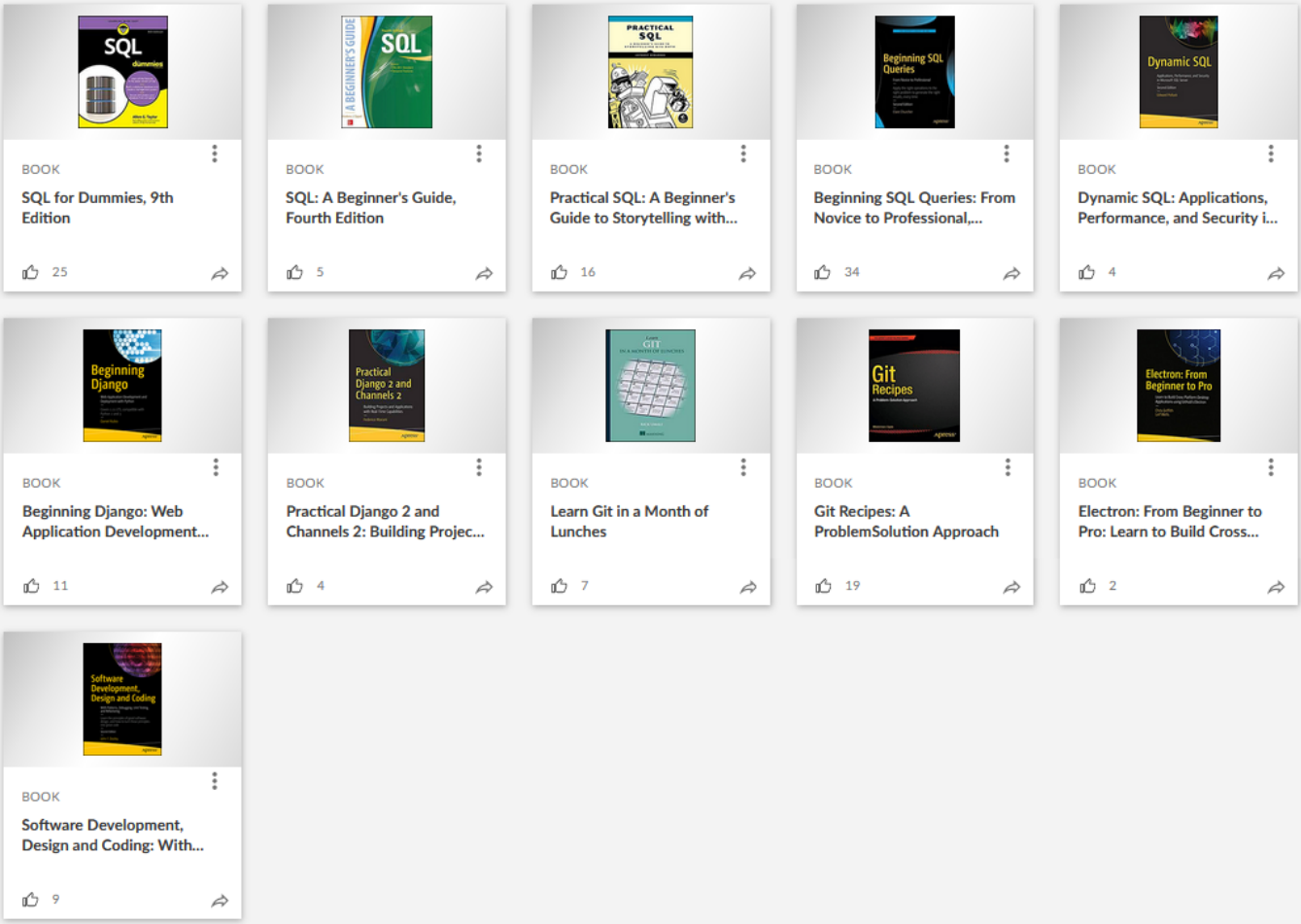
6

BOOK

Joe Celko's SQL for Smarties: Advanced SQL Programmin...

4

Bookshelf Continued



FOLLOW US ON:

www.skilltech.pl

email: biuro@skilltech.pl

tel. +48 22 44 88 827

SkillTech
Technology hired for excellence