FULLSTACK

PROFESSIONAL

Full Stack Professional

CURRICULUM

SPRING BOOT 3

SERVLETS, BEANS,
APPLICATION PROPERTIES,
CONFIG MANAGEMENT

API / HTTP

HTTP, REST AND API
DESIGN BEST PRACTICES

DEVELOPER TOOLS

INTELLIJ ULTIMATE,
TERMINAL,
POSTMAN.
GIT & GITHUB

ERROR HANDLING

CUSTOM EXCEPTIONS
HTTP STATUS CODE

DATABASES & POSTGRESQL

RELATIONAL DATABASE
AND CRUD

SPRING DATA JPA

ENTITIES,
CRUD REPOSITORIES,
CUSTOM QUERIES

FLYWAY

SCHEMA VERSIONING
AUTOMATED DATABASE
MIGHTATIONS

JDBC

DATA SOURCES,
CONNECTION POOL
JDBC TEMPLATES & CRUD

TESTING

UNIT TESTING,
INTEGRATION TESTING
MOCKING & TEST
CONTAINERS

DOCKER

BASICS OF DOCKER
AND CONTAINERS.
DOCKER COMPOSE

AWS

ELASTIC BEANSTALK, ECS, EC2,
LOAD BALANCERS, RDS,
ROUTE 53 & CERTIFICATE
MANAGER

DEVOPS

CONTINUOUS DEPLOYMENT
& DELIVERY WITH
GITHUB ACTIONS

JAVASCRIPT

UP AND RUNNING WITH
JAVASCRIPT

REACT

HOOKS
COMPONENTS
DATA FETCHING
CONTEXT & ROUTING

SPRING SECURITY 6

SPRING SECURITY
ARCHITECTURE,
JWT

LOGIN/REGISTRATION

USER LOGIN AND
REGISTRATION,
PROTECTED ROUTES

TYPESCRIPT

UP AND RUNNING WITH
TYPESCRIPT

ANGULAR

COMPONENTS
ROUTING
DATA FETCHING

AMIGOS CODE

https://amigoscode.com

WHO IS THIS COURSE FOR

Are you Beginner / Junior Dev/ Student?

JAVA MASTER CLASS — 25hrs

From strong fundamentals to advanced concepts, practical experience and project development

Do first!

Mid, Senior & Bootcamp

AMIGOS CODE

https://amigoscode.com

Full Stack Professional

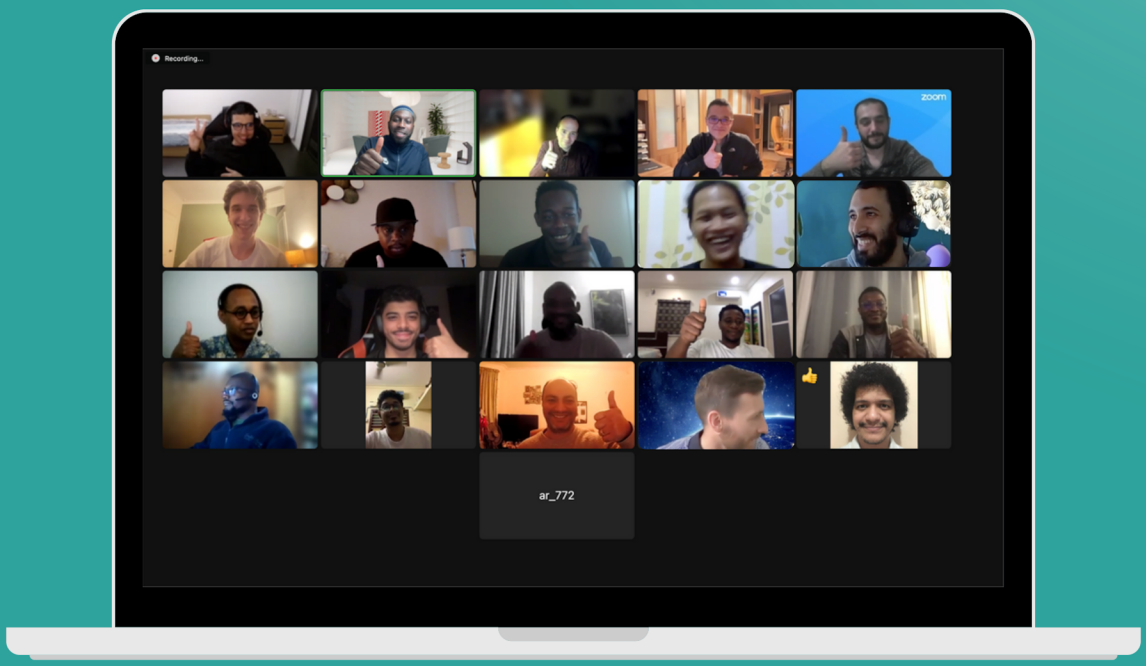
PLUS

MENTORSHIP

ONE-ON-ONE SESSIONS X 2

+

10-WEEK GROUP PROJECT



AMIGOS CODE

https://amigoscode.com

6 Months
INTELLIJ IDEA

AMIGOS CODE

https://amigoscode.com

SPRING BOOT 3

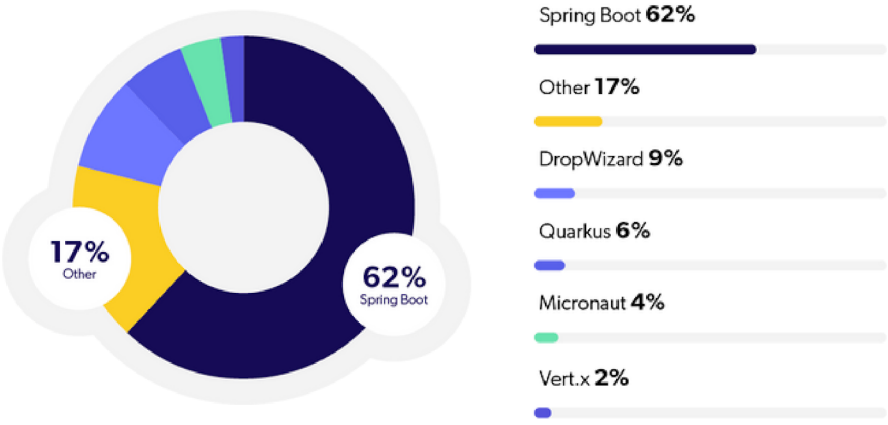
Spring Boot is a Java-based framework used to build production-ready, stand-alone applications. It provides a pre-configured, opinionated platform for building modern, cloud-native applications and microservices. It offers a quick way to create a new project with the minimum required dependencies and configurations reducing the time and effort required to set up a new project from scratch.

Spring Boot also offers several features and tools for managing and deploying applications, including an embedded application server, support for data access and security, and a range of integrations with other popular libraries and frameworks.

SKILLS UPON COMPLETION

- Learn the **Spring Framework** to build scalable web applications that follow industry best practices
- Use **dependency injection** to build modular and maintainable applications that are easy to test and evolve
- Master **backend structuring** techniques to build flexible and extensible architectures
- Build secured REST APIs with **login and registration endpoints** using Spring Boot and JWT authentication to follow industry best practices for security and scalability

MARKET SNEAK PEAK



Source: 2021 Java Developer Productivity Report

HTTP & API

HTTP is the most widely used protocol for data transfer on the web and is an essential component of the internet. It is used by web browsers, servers, and applications to transfer information and enable communication.

APIs, or Application Programming Interfaces, are sets of protocols, routines, and tools for building software applications. They provide a way for different software systems to communicate with each other and exchange data and functionality. An API defines how various software components should interact and the data format that should be used for communication.

SKILLS UPON COMPLETION

- You will gain a **comprehensive understanding** of RESTful APIs and how to implement them using Spring Boot
- You will learn about the **best practices** for API design and how to create APIs that are easy to use, scalable, and maintainable, using Spring Boot's annotations and features.
- Learn how to use the **GET** method to retrieve data from a RESTful API and how to implement it properly to follow industry standards.
- Understand how to use the **POST** method to create data and how to validate user input to ensure proper handling of requests and security.
- Master techniques for using the **PUT** method to update resources in a RESTful API and how to handle errors and conflicts properly.
- Learn how to use the **DELETE** method to remove resources from a RESTful API and how to handle various scenarios like validation, error handling, and security concerns.

DID YOU KNOW?

93.4% OF API DEVELOPERS ARE USING REST

83% OF ALL INTERNET TRAFFIC BELONGS TO API-BASED SERVICES

AMIGOS CODE

https://amigoscode.com

DEVELOPER TOOLS

IntelliJ IDEA Ultimate is a popular Java-integrated development environment (IDE) for developers. It offers a wide range of features, including code editing, debugging, and testing, that help developers to write and manage code more efficiently and a user-friendly interface, making it easy for developers to navigate complex projects and identify potential issues.

The Terminal is a powerful tool that allows developers to interact with the operating system and execute command-line instructions. It enables developers to automate repetitive tasks, manage files and directories, and run scripts and programs.

Postman is a popular API development tool that allows developers to test and debug API requests and responses. It enables developers to easily test the functionality of their APIs, without having to write any code. It is widely used by developers, especially in the API development and testing phase, to ensure that their APIs are functioning as expected.

GitHub is a web-based platform that provides hosting for software development version control and collaboration. It is a crucial tool for developers, as it allows them to manage and store their code repositories in a centralized location and collaborate with other developers on projects.

SKILLS UPON COMPLETION

- You will learn how to use these tools **effectively and efficiently** to write, test, and debug high-quality software, collaborate with other developers, and manage your projects from start to finish.
- You will also learn how to choose the **right tool for the job**, and how to integrate different tools together to build a complete software development environment that meets your specific needs and requirements.

ERROR HANDLING

Error handling is an important aspect of software development, as it helps to ensure that applications respond appropriately in the event of an error. There are two key techniques for handling errors: custom exceptions and HTTP status codes.

Custom exceptions are user-defined exceptions that are thrown by an application in response to a specific error condition. Custom exceptions allow developers to create specific error messages that are relevant to the context of their application. They provide a way to handle specific errors in a controlled manner, allowing developers to identify and resolve problems quickly and efficiently.

HTTP status codes are standardized codes that are used to indicate the outcome of an HTTP request. They are sent by a server in response to a client request and provide information about the success or failure of the request and allow clients to respond appropriately.

SKILLS UPON COMPLETION

- In this course, you will learn how to properly handle exceptions on the server, and **return meaningful HTTP status codes** to clients when errors occur.
- You will learn how to design a **robust and fault-tolerant** server-side architecture, and how to **catch** and handle different types of exceptions, from low-level system errors to business logic exceptions.
- You will also learn how to choose the appropriate HTTP status code for each type of exception, and how to provide detailed error messages that **help clients identify and fix issues** in their own code.
- By mastering exception handling and HTTP status codes, you will be able to create reliable and predictable RESTful APIs that **provide a great user experience** and are **easy to maintain and extend over time.**

DATABASES & POSTGRESQL

Databases are structured collections of data that allow for the storage, retrieval, and manipulation of information. One of the most common types of databases is the **Relational Database**, which organizes data into tables with rows and columns and uses a system of keys to establishing relationships between them.

PostgreSQL is an open-source relational database management system that is known for its reliability, feature-richness, and strong community support. It implements the SQL language for querying and manipulating data and also supports additional advanced data types and functionality.

SKILLS UPON COMPLETION

- You will learn how to perform common **CRUD** (Create, Read, Update, Delete) queries using SQL.
- In addition, you will learn how to work with sequences, a special type of object in PostgresSQL that is often used to generate **unique identifier values** for primary keys.
- You will also become familiar with psql, a **command-line interface (CLI)** client for PostgreSQL that allows you to interact with the database and perform common tasks such as creating tables, modifying data, and running queries.
- By the end of this section, you will have a solid understanding of how to use PostgresSQL to **build** high-performance, scalable, and reliable applications that can handle large amounts of data and provide a great user experience.

DID YOU KNOW?

POSTGRESQL HAS BUILT-IN SUPPORT FOR CONCURRENCY CONTROL, ALLOWING MULTIPLE USERS TO ACCESS AND MODIFY THE SAME DATABASE WITHOUT CONFLICTS OR ERRORS

THIS MAKES IT A POPULAR CHOICE FOR HIGH-TRAFFIC WEB APPLICATIONS AND OTHER SYSTEMS THAT REQUIRE ROBUST AND RELIABLE DATABASE MANAGEMENT.

SPRING DATA JPA

Spring Data JPA is a part of the Spring Framework that provides a way to easily access and manage databases through Java Persistence API (JPA) interfaces.

Entities: A class in Java that represents a table in a database is called an Entity. It has fields that correspond to columns in the table and can be annotated with JPA annotations to specify how it should be mapped to the database.

CRUD Repositories: Spring Data JPA provides a simple way to implement basic CRUD operations on entities through its repository interface. You can extend a repository interface provided by Spring Data JPA to perform basic operations such as save, find, delete and so on.

Custom Queries: You can also write custom queries in JPA to perform more complex operations on entities. You can define these queries using the JPA Query Language (JPQL) or the Criteria API. In addition, you can use the @Query annotation to declare these custom queries directly in your repository interface.

SKILLS UPON COMPLETION

- Learn how to map Java classes to database tables with columns and constraints to create and **manage a database schema** in Spring Boot applications.
- Understand how to use Spring Data JPA interfaces to perform common CRUD (Create, Read, Update, Delete) operations on entities **without the need for boilerplate code**.
- Discover how to create custom queries using JPQL (Java Persistence Query Language) to **handle complex database queries** in your Spring Boot application with ease.

FLYWAY

Flyway is an open-source database migration tool that automates database changes and provides version control for databases. It simplifies the process of maintaining and evolving your database schema, avoids manual errors, and ensures that your database is always in the desired state. With Flyway, you can track changes made to the schema over time, apply plain SQL scripts in numbered migrations, and customize the tool to suit your needs.

SKILLS UPON COMPLETION

- Learn how to use Flyway, a popular open-source database migration tool, to **manage** database schema changes **in a structured and automated way**.
- Understand how Flyway works by **creating and executing database migrations** in a specific order based on version numbers.
- Create **your own database migrations** using SQL scripts that define the changes to your database schema, including creating tables, adding or modifying columns, and defining indexes and constraints.
- Learn about checksums and how Flyway uses them to ensure that your migrations are applied in the **correct order and only once**, even when working with multiple developers and environments.
- **Practice using Flyway with a real database** in the cloud to see how it can help you manage database schema changes effectively and reduce the risk of errors and downtime.

DID YOU KNOW?

Flyway was designed to work with a variety of databases, including SQL Server, Oracle, MySQL, MariaDB, PostgreSQL and others, making it a versatile tool for developers who work with multiple database systems.

JDBC

JDBC (Java Database Connectivity) is a Java API that enables programmers to connect and interact with databases. It provides a set of classes and methods that allow developers to execute SQL statements, retrieve and modify data, and manage database connections within Java applications. By using JDBC, programmers can easily integrate database operations into their Java programs and create dynamic, data-driven applications.

SKILLS UPON COMPLETION

- Learn how to use JDBC, the Java Database Connectivity API, to **interact with relational databases** from your Java applications.
- Explore how JDBC works in conjunction with data sources and connection pools to **manage database connections efficiently and securely**.
- Discover the benefits of JdbcTemplate, a powerful library that provides a simplified and intuitive API for executing SQL queries and mapping the results to Java objects.
- Practice writing SQL queries with JdbcTemplate, and learn how to take advantage of its features such as query parameters, named parameters, and batch updates.
- Gain a deep understanding of how to **map SQL records** to Java records manually, and learn how to work with result sets, prepared statements, and stored procedures.
- By the end of this chapter, you will have the knowledge and skills needed to work with JDBC and JdbcTemplate to build powerful, data-driven applications that can interact with various relational databases.

TESTING

Testing plays a crucial role in software development to ensure code quality and reliability. Various types of testing, including unit testing, integration testing, mocking, and test containers, can be performed to verify that the code functions correctly and handles various inputs and scenarios as expected. Unit tests focus on individual code units, while integration tests check the interaction between different code parts. Mocking isolates code units to simulate dependencies' behaviour, and test containers provide a controlled environment to run tests. Testing enables developers to catch and fix bugs early in the development process, ensuring high-quality software delivery.

SKILLS UPON COMPLETION

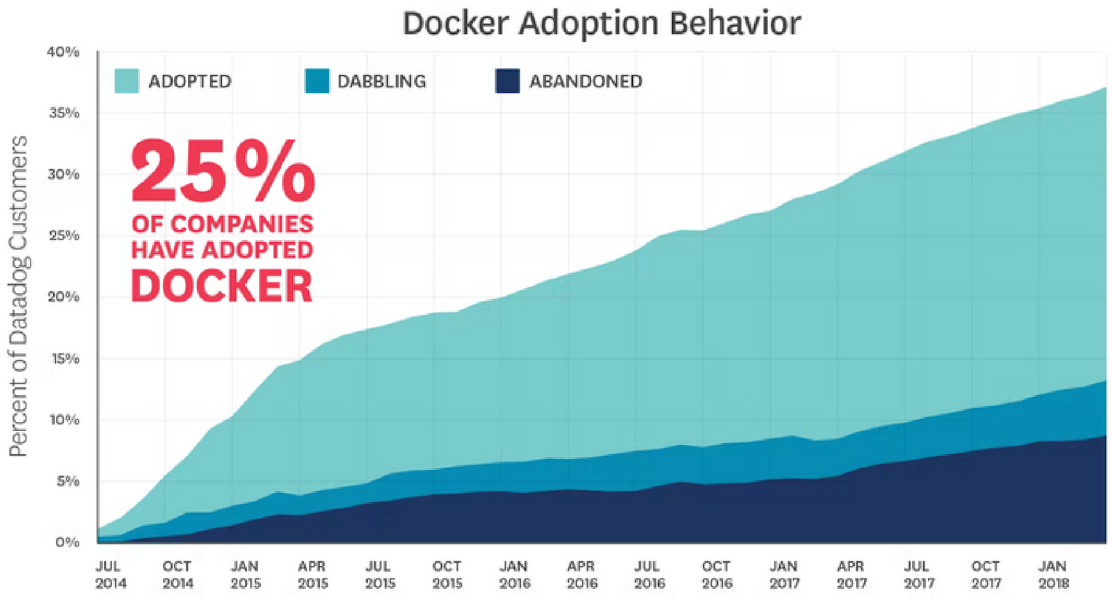
- Learn how to use mocking frameworks like Mockito to simulate the behaviour of external dependencies and isolate the code you want to test. This enables you to test your code in isolation and **avoid dependencies on external services or databases** that can slow down your tests and make them more brittle.
- Understand the importance of testing business logic and how it differs from testing at the UI or API level. Learn how to write tests that validate business rules and requirements, and how to **verify that your code behaves as expected** under various conditions and scenarios.
- Explore the difference between unit and integration tests and how they complement each other to form a **comprehensive testing strategy**. Learn how to write unit tests for individual methods or classes and integration tests that validate the behaviour of your application as a whole, including interactions with databases, external services, and other systems.
- Learn how to use Testcontainers to spin up real, isolated, and disposable instances of external dependencies such as databases, message brokers, and web services for integration testing. Testcontainers make it easy to write **tests that run consistently across different environments and configurations** and help you avoid issues with shared development environments or pre-provisioned test environments that can differ from production.

DOCKER

Docker is a tool that packages an application and its dependencies into a container, providing an isolated and consistent environment for running applications. Containers are isolated from the host system and each other, simplifying the process of developing, deploying, and running applications. Using Docker enables you to move applications between different environments and scale them as needed. Docker Compose helps manage multi-container applications, enabling efficient delivery of complex applications with less effort.

SKILLS UPON COMPLETION

- Understand the basics of containerization and learn what Docker is, how it works, and its advantages for software development and deployment.
- Learn how to work with containers and images, the **fundamental building blocks of Docker**. Understand how they are created, used, and managed, and how to build and customise your own images to meet the specific requirements of your application.
- Master the process of building Docker images and learn **best practices** for creating efficient, secure, and reliable images. Understand how to optimise your images and reduce their size, while ensuring that they meet the needs of your application and infrastructure.
- Explore the process of **deploying and running containers on the cloud**. Learn how to create, manage, and scale containers and container clusters, and how to deploy them to popular cloud platforms such as AWS.



AWS

Amazon Web Services (AWS) is a cloud computing platform that offers a vast array of services, including computing, storage, databases, analytics, machine learning, and much more. It is considered one of the most reliable and scalable cloud platforms in the market, making it a popular choice for businesses and organisations of all sizes. AWS provides benefits such as agility, scalability, and cost-effectiveness, making it an essential tool for those looking to expand their business or optimise their IT operations.

SKILLS UPON COMPLETION

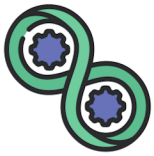
- Learn about Amazon Web Services (AWS) and its cloud computing offerings
- Use Elastic Beanstalk to **deploy and manage** web applications on AWS
- Set up and **manage containers** on AWS with Elastic Container Service (ECS)
- Utilize Amazon RDS to **manage relational databases** on AWS
- Learn about AWS Amplify and how it can be used for building and deploying **mobile and web applications**
- Configure and manage Security Groups to provide network security for your applications on AWS
- Use Application Load Balancers, Listeners, Target Groups, and Health Checks to distribute traffic across instances and ensure high availability and reliability for your applications.

DID YOU KNOW?

AWS offers over 165 fully managed services, making it one of the most comprehensive cloud computing platforms available.

This includes services for: computing, storage, databases, analytics, machine learning, security, mobile, and more, allowing organizations of all sizes to build and run their applications and services in the cloud.

DEV-OPS



DevOps is a software development approach that emphasizes collaboration and communication between development and operations teams to deliver high-quality software more efficiently. A key component of DevOps is CI/CD, which automates the software delivery process by integrating code changes frequently and deploying them automatically to production.

By implementing CI/CD, developers and operations teams can work more closely together to improve the software delivery process. Continuous Integration ensures that code changes are frequently integrated, allowing developers to catch and fix bugs early on. Continuous Deployment automates the deployment of code changes, reducing the time and effort required to deploy updates to production.

SKILLS UPON COMPLETION

- GitHub Actions: **Automate the development process** with powerful workflows and integrations, all within the familiar GitHub environment.
- Workflows, Actions, Jobs, Steps and Secrets: Learn the key concepts that form the **basis of a successful GitHub Actions workflow**, including how to define and structure jobs, steps and secrets.
- Continuous Integration and Delivery: Master the skills needed to implement a continuous integration and delivery process that ensures high-quality, reliable software is delivered to production with minimal risk.
- Slack Webhooks: Leverage the power of Slack to create c**ustomised notifications and alerts**, and integrate them seamlessly into your workflows.

DID YOU KNOW

CI/CD, developers can detect and resolve integration issues up to 30x faster compared to traditional development methods?

JAVASCRIPT

JavaScript is a high-level, interpreted programming language used to create dynamic and interactive web pages. It supports different programming styles, including event-driven, functional, and imperative programming. JavaScript is an object-oriented language that is executed on the client side within the user's web browser. This unique feature makes it a crucial tool for web developers in creating interactive and engaging web experiences for their users.

SKILLS UPON COMPLETION

By mastering the fundamentals of JavaScript, you will be better equipped to build more advanced and feature-rich applications with React, as well as better understand the underlying principles and concepts that make React so powerful.

- Learn how to work with variables, functions, and loops to create more dynamic and interactive JavaScript applications.
- Master **advanced techniques** like callback functions, map functions, and object destructuring to write more efficient and effective code.
- Explore the **power of modules** and how they can help you organise your code and create reusable components.
- Dive into the world of Promises and Async Await to handle asynchronous JavaScript and **create more responsive and scalable applications.**

DID YOU KNOW?

JavaScript is an event-driven programming language, meaning that it can respond to events such as user interactions and manipulate the content of a web page in real-time. This makes it a powerful tool for creating dynamic and interactive user experiences, and it has paved the way for the development of advanced web applications and single-page applications (SPAs).

REACT

React is a popular JavaScript library for building user interfaces, commonly used for developing single-page applications and mobile applications. It utilises a virtual Document Object Model (DOM) to efficiently render components and update the UI in response to changes in data. React components are defined using a declarative syntax, making it easy to understand and debug code. This library also supports reactive and functional programming, enabling developers to write clean, maintainable, and scalable code. With these features, React offers an efficient and powerful toolset for developers to create modern, responsive, and dynamic web applications.

SKILLS UPON COMPLETION

Build by!

You will build a complete React application that includes a login and registration system with protected routes, which communicates with a REST API. By the end of the chapter, you will have gained an understanding of the following topics:

- Components: You will learn how to use components to build **modular and reusable pieces of code**, making it easy to build complex user interfaces.
- Hooks: You will learn how to use hooks to **manage state and lifecycle methods** within your React components, allowing for more streamlined code and better performance.
- Data Fetching: You will learn how to **fetch data from external sources** using various methods such as Axios or fetch API, making your React applications more dynamic and interactive.
- React Context: You will learn how to use the React Context API to manage global state, allowing you to easily **share data between components** without having to pass it down manually through props.
- Click Events: You will learn how to handle click events within your React components, allowing you to **build more interactive and engaging** user interfaces.
- Working with Forms: You will learn how to build forms in React and manage form data, including form validation and error handling.
- Local Storage: You will learn how to use local storage to persist data on the client side, enabling a **seamless user experience** across sessions.

SPRING SECURITY 6

Spring Security is a framework for securing Java-based web applications, providing a powerful and customizable set of security features such as authentication, authorization, and access control. As part of the Spring Framework, it is widely used for securing both traditional web applications and microservices-based architectures. With Spring Security, developers can easily control who has access to what parts of their application, by defining security rules in code or XML-based configuration files. This enables them to create secure and dynamic applications that meet the specific needs of their users.

SKILLS UPON COMPLETION

- Learn to secure your Java web applications using Spring Security architecture and JSON Web Tokens (JWT), which can help **protect your users' data** and provide peace of mind.
- Implement custom filters and filter chains to tailor the security of your application to your specific needs, allowing you to create a security model that is **customized for your application and its users**.
- Configure and work with the Authentication Manager and Authentication Providers to manage user authentication and authorization, giving you the ability to **manage user access and verify their identities with ease.**
- Gain a deeper understanding of Security Configuration, enabling you to provide a secure environment for your application's users, which can help you meet **regulatory requirements and build trust with your users**.

DID YOU KNOW?

Spring Security offers a flexible and modular approach to security that allows developers to easily customize their security configurations based on their specific requirements.

LOGIN/REGISTRATION

User login and registration enable users to create an account and access protected content. The application stores user credentials securely in a database and uses them to verify the user's identity when they log in. Protected routes in the application check the user's authentication status and redirect them to a login page if they are not logged in. This ensures the security of the application and its data.

SKILLS UPON COMPLETION

In this chapter, you will learn how to build two **essential components of a web application**: user registration and login functionality. These are the **first points of contact** with your users and require careful attention to security and usability. You will build these applications using two of the most popular front-end frameworks, React and Angular.

- You will gain experience in building web applications with these powerful frameworks and learn how to work with their respective **toolsets and best practices**.
- you will be able to use authentication and **authorization concepts**, including storing user data securely and managing user sessions.
- By the end of this chapter, you will have the skills to build robust and secure user authentication systems for **your own web applications**.

DID YOU KNOW?

JSON Web Tokens (JWT) are a widely used standard for transmitting information securely as a JSON object between parties in web applications, especially in stateless authentication systems.

JWTs can be signed, encrypted, or both, and they allow for the transfer of claims, such as user identity or permissions, between systems in a compact and secure format.

TYPESCRIPT

TypeScript is an object-oriented language that adds optional type annotations, class-based OOP, and interfaces to JavaScript. Developed and maintained by Microsoft, it's popular for building complex applications for the web and desktop. TypeScript's strong typing features make it suitable for developing enterprise applications and popular for front-end development using frameworks like Angular.

SKILLS UPON COMPLETION

- You will learn the **basics of TypeScript**, including its syntax, data types, and variables, to enable you to start building Angular applications.
- TypeScript offers a range of data types, including primitive types, user-defined types, and advanced types such as union and intersection types. You will learn how to use these types to **ensure your code is both type-safe and robust**.
- TypeScript extends the object-oriented features of JavaScript with classes and interfaces, enabling you to write more **maintainable and scalable code**. You will learn how to define and use classes and interfaces, as well as inheritance, access modifiers, and static properties.
- TypeScript's generic type feature allows you to write reusable, type-safe code that works with a variety of data types. You will learn how to use generics in your code, including function and class generics, to **improve the readability and maintainability of your code.**

DID YOU KNOW?

TypeScript is a statically typed superset of JavaScript, meaning that it extends the capabilities of JavaScript with optional type annotations.

This allows developers to catch type-related errors at compile time, instead of at runtime, which can help to prevent bugs and improve the maintainability of their code.

ANGULAR

Angular is a popular open-source web application framework that is widely used for building complex and dynamic client-side applications. Developed and maintained by Google, it offers a component-based architecture, a powerful template system, and advanced features such as dependency injection and change detection. Angular also provides a strong community, a rich ecosystem of libraries, and comprehensive documentation, making it a great choice for developers looking to build high-quality, scalable, and maintainable web applications.

SKILLS UPON COMPLETION

- You will learn about the overall **architecture of an Angular application**, including how to organize your code, how the different parts of an Angular app work together, and best practices for developing an Angular app.
- Use type and event binding to **make your components dynamic and interactive**. Learn how to create and use components, which are the building blocks of Angular applications.
- **Protect specific routes** and ensure that only authenticated users can access certain parts of your app. Use the Angular router to create multiple views in your application.
- Learn how to share data and functionality between different parts of your application using services. Use HTTP services to communicate with a REST API to **retrieve and store data**.
- Fetch data from a REST API using Angular's built-in HTTP client. Learn how to **handle errors and display data** to the user in a user-friendly way.
- Learn how to create and validate forms, and use Angular's built-in form controls and directives to **make working with forms easier**.

DID YOU KNOW?

Angular uses a component-based architecture, which allows developers to build applications as a set of interconnected components, each with its own template, logic, and styles.