

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi, Karnataka – 590018



INTERNSHIP REPORT

ON

“WEB DEVELOPMENT”

Submitted on partial fulfilment of academic requirement of Final year

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

MOHAMMED JABIR (1VJ19CS034)

Carried out at HQV Technologies Private Limited
No 17, Third Floor C Street, Bharathi Nagar, Bangalore – 560001

Internal Guide

Dr. Naveen G

External Guide

Mr. Mohammed Viqar Ahmed
Director HQV Technologies

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VIJAYA VITTALA INSTITUTE OF TECHNOLOGY

BANGALORE KARNATAKA,

INDIA -560077

2022-23

VIJAYA VITTALA INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING



CERTIFICATE

This is to certify that the project entitled **“WEB DEVELOPMENT”** is a Bonafide work carried out by **MOHAMMED JABIR (1VJ19CS034)** in partial fulfillment for the award of degree in Bachelor of Engineering in Computer Science Engineering from Visvesvaraya Technological University, Belagavi during the academic year 2022-23. It is certified that all the corrections/suggestions indicated for internal assessment have been incorporated in the report submitted in the department Library. This Project report has been approved as it satisfies the academic requirements in respect of Project report prescribed for award of said degree.

.....

Signature of Internal Guide

Dr. Naveen G

Professor

Dept. of CSE

.....

Signature of HOD

Pof. Mangala M Patil

Prof. and Head

Dept. of CSE

.....

Signature of Principal

Dr.S Rajendra

Principal

Name of the Examiners

1.....

2.....

Signature with Date

1.....

2.....

DECLARATION

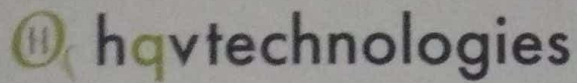
I, **MOHAMMED JABIR** student of 8th semester in Computer Science and Engineering, Vijaya Vittala Institute of Technology, Bengaluru, hereby declare the internship training titled “**Web Development**” has been carried out by me under the supervision and guidance of **Mr. Mohammed Viqar Ahmed**, HQV Technologies and **Prof. Naveen G** Assistant Professor Dept. of CSE Vijaya Vittala Institute of Technology, Bengaluru and submitted in partial fulfillment for the award of degree in Bachelor of Engineering in Information Science and Engineering of Visvesvaraya Technological University, Belagavi during the academic year 2022 - 2023. I further declare that thereport has not been submitted to any other University for the award of any other degree.

Place: Bangalore

Date

MOHAMMED JABIR (1VJ19CS034)

OFFER LETTER PROVIDED BY THE COMPANY



Innovation lab for research and development

20th August, 2022

Bangalore

Dear **Mohammed Jabir**,

We are delighted to welcome you to HQV Technologies Private Limited as a **Web Developer Intern**. This internship is for 40 days starting from 22nd Aug 2022. At HQV Technologies Private Limited, we believe that our quality work is our biggest strength.

After completion of the internship, you will be provided with an Internship Completion Certificate.

We wish you all the very best for your very first step toward your job career.

Thanking You

From HQV Technologies Private Limited



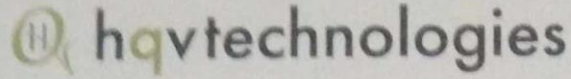
Mohammed Viqar Ahmed
Director
Mobile No +91 8892614271

HQV Technologies Private Limited

Website: hqvtechnologies.com

No 17, Third Floor C Street, Bharathi Nagar, Bangalore – 560 001.

CERTIFICATE OF INTERNSHIP COMPLETION



Innovation lab for research and development

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Mr. Mohammed Jabir (1VJ19C5034)** a student at **Vijaya Vittala Institute of Technology** has successfully completed his internship at **HQV Technologies Private Limited** from **21st August 2022** to **1st October 2022**. He worked as an intern in application development team using **agile methodology**. During his internship he demonstrated good technical skills and carried out the tasks assigned to him proficiently. We wish him all success on his future assignments.

Areas worked on: Backend (Rest API, RDBMS), UI Mocks.

Mohammed Viqar Ahmed
3/10/2022

Mohammed Viqar Ahmed,
Director.
Mobile No. +91 8892614271



HQV Technologies Private Limited

Website: hqvtechnologies.com

No 17, Third Floor C Street, Bharathi Nagar, Bangalore – 560 001.

ACKNOWLEDGEMENT

I consider it a privilege to whole-heartedly express our gratitude and respect to each and every one who guided and helped us in the successful completion of these projects.

I am grateful to Principal, Dr. S Rajendra, VVIT, Bangalore and all staff members of Computer Science and Engineering Department for their kind co-operation.

I am extremely thankful to Pro. Mangala M Patil, Professor and HOD, Department of Computer Science and Engineering, for his co-operation and encouragement. I thank him for providing me an opportunity to carry out the project at college.

I express our deepest gratitude and sincere thanks to our guide Pro. Kavita C, Assistant professor, Department of Computer Science and Engineering for his valuable guidance during the course of this Project and continuous suggestions to make our Project successful.

I wish to express my heartfelt gratitude to my external guide Mr. Mohammed Viqar Ahmed, Director of HQV Technologies, Bengaluru for his valuable guidance, suggestions and cheerful encouragement during the entire period of my Internship Training.

Finally, it is a pleasure and happiness to the friendly cooperation showed by all the staff members of Computer science engineering department.

MOHAMMED JABIR (1VJ19CS034)

ABSTRACT

I was offered an internship at HQV Technologies Private Limited after successful round of interview. The internship was for 40 days from 21-Aug-2022 to 01-Oct-2022. The role, I was assigned was that of a web developer.

The study aimed to analyze, to design, to implement, to test some Backend (Rest API, RDBMS) like location API's for an ongoing project, I learnt a lot about the procedures used in the industry and the techniques to work on a project like this. This location API's are intended to be used in a project for finding the bed availability in hospitals in a particular area.

TABLE OF CONTENT

| Sl no. | TOPIC | Page no. |
|--------|---|----------|
| 1. | Chapter 1 Company Profile | 1 |
| 2. | Chapter 2 Introduction | 2-3 |
| | 2.1 Introduction to Internship | |
| | 2.2 Internship Program objective | 2 |
| | 2.2 About The Company | 2 |
| | 2.3.1 Vision | 3 |
| | 2.3.2 Mission | 3 |
| | 2.3.3 Value | 3 |
| 3. | Chapter 3 System Requirements and System Analysis | 4 |
| | 3.1 System Requirements | 4 |
| | 3.1.1 Functional Requirements | 4 |
| | 3.1.2 Non-Functional Requirements | 4 |
| | 3.1.3 System Requirements Specification | 4 |
| 4. | Chapter 4 Task Performed | 5-25 |
| | 4.1 Object Oriented Programming | 5 |
| | 4.1.1 Class | 5 |
| | 4.1.2 Object | 5 |
| | 4.1.3 Method | 6 |
| | 4.1.4 Pillars of OOPs | 6 |
| | 4.1.4.1 Abstraction | 6 |
| | 4.1.4.2 Encapsulation | 7 |
| | 4.1.4.3 Polymorphism | 8 |
| | 4.2 Git | 9 |
| | 4.2.1 Working of GitHub | 9 |
| | 4.2.2 Git Commands | 10 |
| | 4.3 UI Mockup | 12 |
| | 4.3.1 Balsamiq | 13 |
| | 4.4 REST API's | 13 |
| | 4.4.1 HTTP Methods for RESTful Services | 13 |
| | 4.5 Spring Boot | 14 |
| | 4.5.1 Building Rest API | 16 |
| | 4.6 Postman | 21 |
| | 4.6.1 Testing API's using Postman | 21 |
| | 4.7 Project | |
| 5. | Chapter 5 Reflection | 26 |
| 6. | Chapter 5 Conclusion | 27 |

LIST OF FIGURES

| Fig No. | TOPIC | Page No |
|----------------|----------------------------|----------------|
| 4.1 | Java Class and Objects | 6 |
| 4.2 | Abstraction | 7 |
| 4.3 | Encapsulation | 8 |
| 4.4 | Spring Initializer | 16 |
| 4.5 | Postman adding tests | 21 |
| 4.6 | Postman adding collections | 22 |
| 4.7 | Country DTO Class | 22 |
| 4.8 | Country Entity Class | 23 |
| 4.9 | Country Service Class | 23 |
| 4.10 | Country Repository Class | 24 |
| 4.11 | Country RestApi Class | 24 |
| 4.12 | Console Output | 25 |
| 4.13 | Country HTTP Response | 26 |

CHAPTER 1

COMPANY PROFILE

| | |
|------------------------------|---|
| Name | HQV Technologies Private Limited |
| Address | No 17, Third Floor C Street, Bharathi Nagar, Bangalore – 560 001 |
| Contact No. | +91 8892614271 |
| Email | www.viqar@hqvtechnologies.com |
| Website | www.hqvtechnologies.com |
| Type of the Company | Private |
| Nature of the company | Information Technology |
| Company logo |  |
| Vision | To be a world-class IT Application Development Training and internship organisation committed to helping students and providing quality services |

CHAPTER 2

INTRODUCTION

2.1 Introduction to Internship

An internship is a structured work experience related to a student's main goals and/or career goals. It is an experience that fosters students' academic, professional, and personal growth. This is a commitment made with an employer that is closely related to the student's major and deems it desirable to support the student's education and training. He is typically short-term, almost 6 weeks, conducted through face-to-face, direct contact hours or a training program designed to enable the intern to develop a summary report of their experience. This internship program is designed by HQV Technologies for the partial completion of a Bachelor of Engineering degree. Interns are recognized through an internship program within this curriculum. The program enhanced the skills and enthusiasm of students as they gained knowledge of the corporate environment and learned different aspects of working mechanisms common in organizations.

2.2 Internship Program Objective

The major objectives of internship are:

- To expose students to a particular job and a profession or industry.
- To provide students with opportunity to develop skills in the field of interest.
- To assist students in gaining vital work-related experience and building strong resume for bright career.

2.3 About the Company

HQV Technologies Private Limited is a Private incorporated on 22 June 2022. It is classified as Non-govt Company and is registered at Registrar of Companies, Bangalore.

HQV Technologies is a research and development center and educational institution based in Bangalore. They focus on providing quality education on the latest technologies and developing much-needed products for society. They operate project consultants who undertake a variety of projects from a variety of clients, assist in designing and manufacturing products, and provide services. They are continuously involved in researching future technologies and finding ways to simplify them for our customers.

2.3.1 Vision

To be a world-class IT Application Development Training and internship organisation committed to helping students and providing quality services.

2.3.2 Mission

To Harness and train best brainpower to give solutions for real challenges of the world.

2.3.3 Values

- Respect for dignity and potential of individuals.
- Enthusiasm for top performance and desire for change .
- Honesty and fairness in everything.
- Ensure speed of response.
- Faster learning, creativity and team-work.
- Loyalty and pride in the company.

CHAPTER 3

SYSTEM REQUIREMENTS AND SYSTEM ANALYSIS

3.1 System Requirements

3.1.1 Functional Requirements

Functional requirement of the system describes what the system does. The main functional requirements of this system are as follows:

- User should be able to view all the necessary information and specification about this project.
- User should have Spring Tool Suite 3 or above installed on the device.
- Browser compatibility.
- User should have Postman installed.

3.1.2 Non-Functional Requirements

A non-functional requirement describes how the system performs a certain function. Non-functional requirements generally specify the system's quality attributes or properties such as reliability, usability, storage occupancy, performance, and response time.

3.1.3 SYSTEM REQUIREMENT SPECIFICATION

- Should describe functional and non-functional requirements so that they are understandable by system users who don't have detailed technical knowledge.
- User requirements are defined using natural language, tables and diagrams.

Software Requirement Specification

- Languages: Java.
- Framework: Spring Tool Suite.
- Testing tool: Postman.

Hardware Requirements Specification

- Processor: Intel core i3 or i5.
- Hard Disk: 5 GB
- Ram: 4 GB

CHAPTER 4

Task Performed

All tasks performed during the internship program were based on backend (Rest API, RDBMS), UI mocks development. The trainers have assigned some basic tasks to industry standards to make the technology easy to understand.

4.1 Object Oriented Programming (OOPs) Concept in Java

4.1.1 Class

A class is a user-defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type. In general, class declarations can include these components, in order:

1. **Modifiers:** A class can be public or has default access (Refer this for details).
2. **Class keyword:** class keyword is used to create a class.
3. **Class name:** The name should begin with an initial letter (capitalized by convention).
4. **Superclass:** The name of the class's parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
5. **Interfaces;** A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
6. **Body:** The class body is surrounded by braces, { }.

Constructors are used for initializing new objects. Fields are variables that provide the state of the class and its objects, and methods are used to implement the behavior of the class and its objects.

There are various types of classes that are used in real time applications such as nestedclasses, anonymous classes, lambda expressions.

4.1.2 Object

It is a basic unit of Object-Oriented Programming and represents real-life entities. A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of :

State: It is represented by attributes of an object. It also reflects the properties of an object.

Behavior: It is represented by the methods of an object. It also reflects the response of an object with other objects.

Identity: It gives a unique name to an object and enables one object to interact with other objects.

Objects correspond to things found in the real world. For example, a graphics program may have objects such as “circle”, “square”, and “menu”. An online shopping system might have objects such as “shopping cart”, “customer”, and “product”.

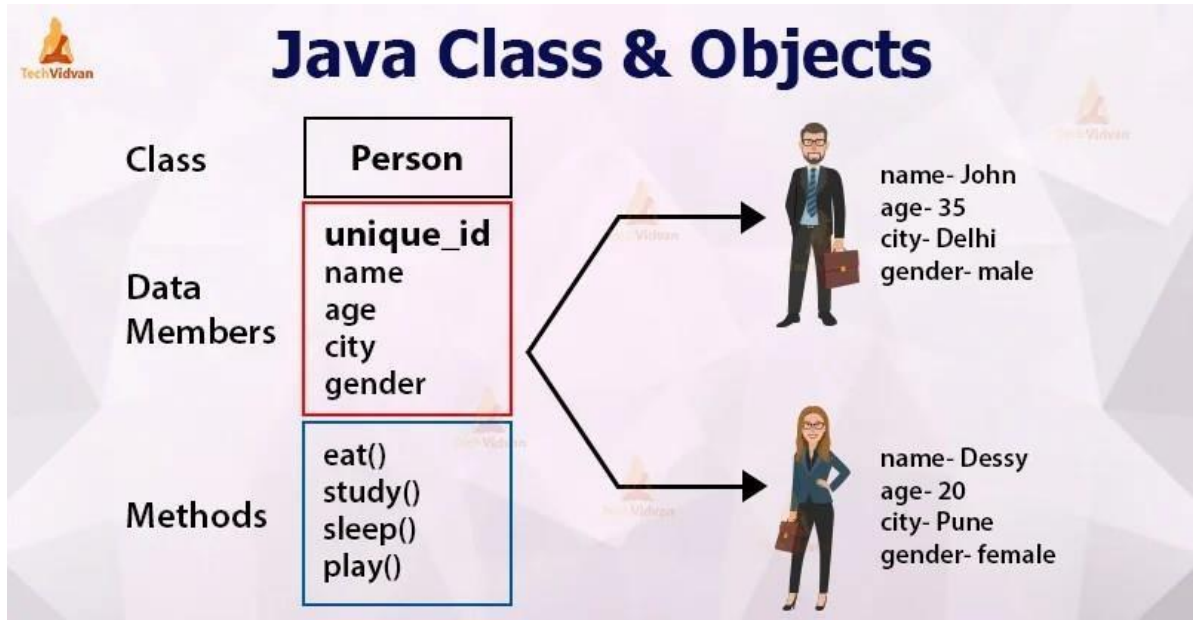


Fig 4.1: Java Class and Objects

4.1.3 Method

Method in Java or Java Method is a collection of statements that perform some specific task and return the result to the caller. A Java method can perform some specific task without returning anything. Methods in Java allow us to **reuse** the code without retyping the code. In Java, every method must be part of some class that is different from languages like C, C++, and Python.

1. A method is like function i.e. used to expose behavior of an object.
2. it is a set of codes that perform a particular task.

Advantage of Method

Code Reusability

Code Optimization

4.1.4 Pillars of OOPs

4.1.4.1 Abstraction

Data Abstraction is the property by virtue of which only the essential details are displayed to the user. The trivial or the non-essential units are not displayed to the user. Ex: A car is viewed as a car rather than its individual components.

Data Abstraction may also be defined as the process of identifying only the required characteristics of an object ignoring the irrelevant details. The properties and behaviors of an object differentiate it from other objects of similar type and also help in classifying/grouping the objects.

Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of a car or applying brakes will stop the car, but he does not know how on pressing the accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of the accelerator, brakes, etc in the car. This is what abstraction is.

In java, abstraction is achieved by interfaces and abstract classes. We can achieve 100% abstraction using interfaces.

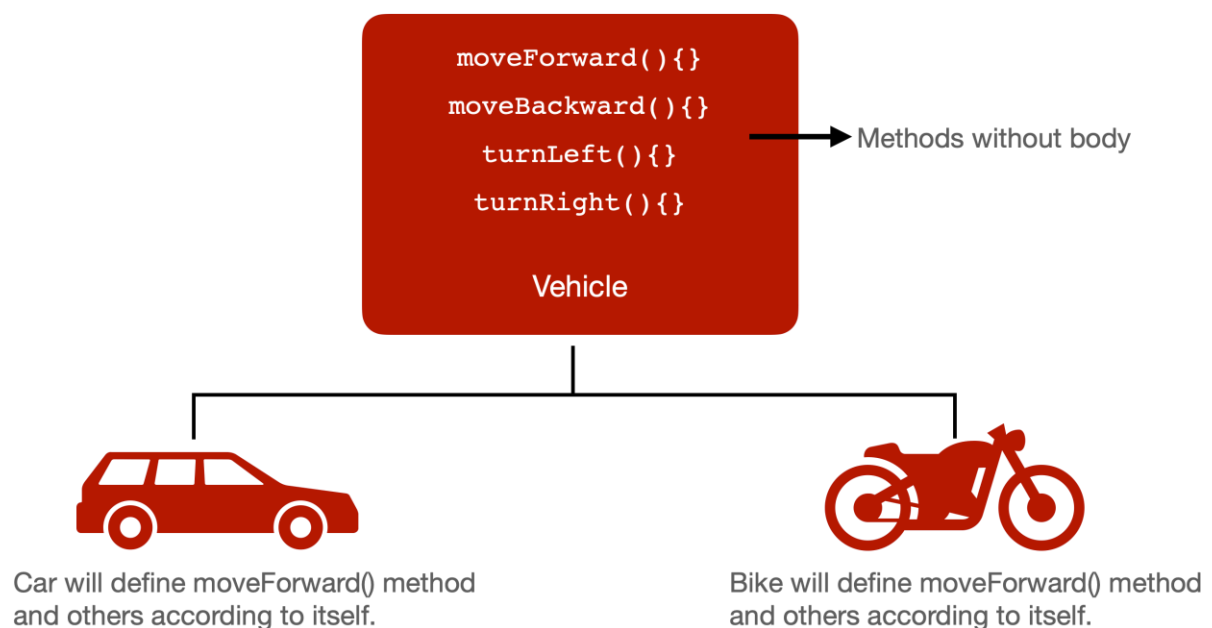


Fig 4.2: Abstraction

4.1.4.2 Encapsulation

Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. Another way to think about encapsulation is, that it is a protective shield that prevents the data from being accessed by the code outside this shield.

Technically in encapsulation, the variables or data of a class is hidden from any other class and can be accessed only through any member function of its own class in which it is declared.

As in encapsulation, the data in a class is hidden from other classes using the data hiding concept which is achieved by making the members or methods of a class private, and the

class is exposed to the end-user or the world without providing any details behind implementation using the abstraction concept, so it is also known as a **combination of data-hiding and abstraction**.

Encapsulation can be achieved by Declaring all the variables in the class as private and writing public methods in the class to set and get the values of variables.

It is more defined with the setter and getter method.

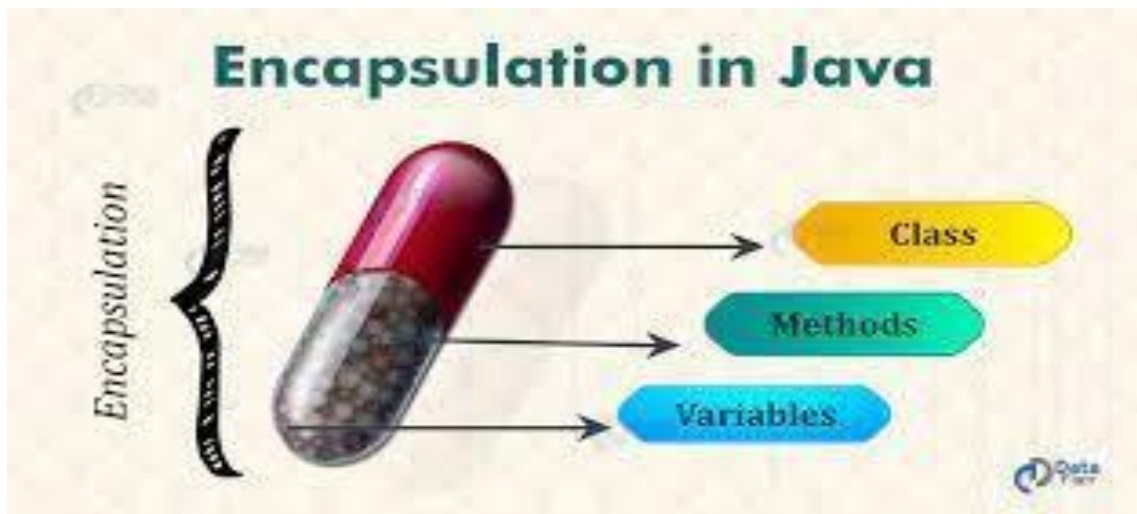


Fig 4.3: Encapsulation

Inheritance is an important pillar of OOP(Object-Oriented Programming). It is the mechanism in java by which one class is allowed to inherit the features(fields and methods) of another class. In Java, inheritance means creating new classes based on existing ones. A class that inherits from another class can reuse the methods and fields of that class. In addition, you can add new fields and methods to your current class as well.

Inheritance in Java: Why do we need it?

The code written in the Superclass is common to all subclasses. Child classes can directly use the parent class code.

Method Overriding is achievable only through Inheritance. It is one of the ways by which java achieves Run Time Polymorphism.

The concept of abstract where we do not have to provide all details is achieved through inheritance. Abstraction only shows the functionality to the user.

4.1.4.2 Polymorphism

The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.

Real-life Illustration: Polymorphism

A person at the same time can have different characteristics. Like a man at the same time is a

father, a husband, an employee. So the same person possesses different behavior in different situations. This is called polymorphism.

Polymorphism is considered one of the important features of Object-Oriented Programming. Polymorphism allows us to perform a single action in different ways. In other words, polymorphism allows you to define one interface and have multiple implementations. The word “poly” means many and “morphs” means forms, So it means many forms.

Types of polymorphism

In Java polymorphism is mainly divided into two types:

Compile-time Polymorphism

Runtime Polymorphism

Type 1: Compile-time polymorphism

It is also known as static polymorphism. This type of polymorphism is achieved by function overloading or operator overloading.

Type 2: Runtime polymorphism

It is also known as Dynamic Method Dispatch. It is a process in which a function call to the overridden method is resolved at Runtime. This type of polymorphism is achieved by Method Overriding. **Method overriding**, on the other hand, occurs when a derived class has a definition for one of the member functions of the base class. That base function is said to be **overridden**.

4.2 GIT

Git is a distributed version control system that tracks changes in any set of computer files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows (thousands of parallel branches running on different systems).

Git is the most widely used version control system in software development, and GitHub leverages this technology for its service, hence its name.

4.2.1 Working of GitHub

GitHub users create accounts, upload files, and create coding projects. But the real work of GitHub happens when users begin to collaborate.

While anyone can code independently, teams of people build most development projects.

asynchronously. There are many challenges to creating collaborative projects with distributed teams. GitHub makes this process much simpler in a few different ways.

First, all the code and documentation are in one place. This limits issues with access for anyone who wants to contribute to a project. Each repository also contains instructions and other details to help outline project goals and rules.

Next, coding is more creative and abstract than most non-technical people think it is. For example, say two developers are working on different pieces of code. These two pieces of code should work together. But sometimes one piece of code can make the other code fail. Or a piece of code can have an unexpected impact on how the other code works.

GitHub solves these problems by showing how both files will change the main branch. It catches these errors before pushing changes, making the coding process more efficient.

GitHub also makes it easier to track changes and go back to previous versions of a project. To explain this, we'll need to understand the technology that GitHub is based on, Git, and talk about version control.

4.2.2 Git Commands

Getting & Creating Projects

| Command | Description |
|--|--|
| <code>git init</code> | Initialize a local Git repository |
| <code>git clone ssh://git@github.com/[username]/[repository-name].git</code> | Create a local copy of a remote repository |

Basic Snapshotting

| Command | Description |
|---|---|
| <code>git status</code> | Check status |
| <code>git add [file-name.txt]</code> | Add a file to the staging area |
| <code>git add -A</code> | Add all new and changed files to the staging area |
| <code>git commit -m "[commit message]"</code> | Commit changes |
| <code>git rm -r [file-name.txt]</code> | Remove a file (or folder) |

Branching & Merging

| Command | Description |
|--|---|
| git branch | List branches (the asterisk denotes the current branch) |
| git branch -a | List all branches (local and remote) |
| git branch [branch name] | Create a new branch |
| git branch -d [branch name] | Delete a branch |
| git push origin --delete [branch name] | Delete a remote branch |
| git checkout -b [branch name] | Create a new branch and switch to it |
| git checkout -b [branch name] origin/[branch name] | Clone a remote branch and switch to it |
| git branch -m [old branch name] [new branch name] | Rename a local branch |
| git checkout [branch name] | Switch to a branch |
| git checkout - | Switch to the branch last checked out |
| git checkout -- [file-name.txt] | Discard changes to a file |
| git merge [branch name] | Merge a branch into the active branch |
| git merge [source branch] [target branch] | Merge a branch into a target branch |
| git stash | Stash changes in a dirty working directory |
| git stash clear | Remove all stashed entries |

Sharing & Updating Projects

| Command | Description |
|--|---|
| git push origin [branch name] | Push a branch to your remote repository |
| git push -u origin [branch name] | Push changes to remote repository (and remember the branch) |
| git push | Push changes to remote repository (remembered branch) |
| git push origin --delete [branch name] | Delete a remote branch |
| git pull | Update local repository to the newest commit |

| Command | Description |
|--|---|
| <code>git pull origin [branch name]</code> | Pull changes from remote repository |
| <code>git remote add origin ssh://git@github.com/[username]/[repository-name].git</code> | Add a remote repository |
| <code>git remote set-url origin ssh://git@github.com/[username]/[repository-name].git</code> | Set a repository's origin branch to SSH |

Inspection & Comparison

| Command | Description |
|---|--------------------------------|
| <code>git log</code> | View changes |
| <code>git log --summary</code> | View changes (detailed) |
| <code>git log --oneline</code> | View changes (briefly) |
| <code>git diff [source branch] [target branch]</code> | Preview changes before merging |

4.3 UI Mockup

A UI mockup is a visual representation of a final digital product or website, including layout/hierarchy, color, typography, icons, and other UI elements. While mockups are high-fidelity designs, they are static and have no functionality-like a screenshot.

Mockups are a crucial part of the design thinking process because they answer important visual questions (like layout, color, and hierarchy) and allow designers to start high-fidelity prototyping.

Mockups also provide engineers with a visual reference to start the development phase. With UXPin, devs can use Spec Mode to analyze each mockup's sizing, spacing, grid/layout, colors, and typography.

Mockups also provide:

- **Meaningful stakeholder feedback:** Thanks to the higher fidelity, mockups require less context than low-fidelity wireframes and sketches, giving stakeholders an accurate representation of the final product.

- **Realistic perspective:** Mockups reveal problems that might not have been obvious during low-fidelity wireframing—like accessibility considerations, poor color choices, or layout issues.
- **Flexibility:** It's easier to make changes to a UI mockup using a design tool than editing code. If designers work with a design system or UI component library, making changes is as easy as swapping components or rearranging the layout.

4.3.1 Balsamiq

Balsamiq Wireframes is a graphical user interface website wireframe builder application. It allows the designer to arrange pre-built widgets using a drag-and-drop WYSIWYG editor. The application is offered in a desktop version as well as a plug-in for Google Drive, Confluence and JIRA.

4.4 REST API's

REST APIs provide a flexible, lightweight way to integrate applications, and have emerged as the most common method for connecting components in microservices architectures.

An API, or *application programming interface*, is a set of rules that define how applications or devices can connect to and communicate with each other. A REST API is an API that conforms to the design principles of the REST, or *representational state transfer* architectural style. For this reason, REST APIs are sometimes referred to RESTful APIs.

First defined in 2000 by computer scientist Dr. Roy Fielding in his doctoral dissertation, REST provides a relatively high level of flexibility and freedom for developers. This flexibility is just one reason why REST APIs have emerged as a common method for connecting components and applications in a microservices architecture.

4.4.1 HTTP Methods for RESTful Services

The HTTP verbs comprise a major portion of our “uniform interface” constraint and provide us the action counterpart to the noun-based resource. The primary or most-commonly-used HTTP verbs (or methods, as they are properly called) are POST, GET, PUT, PATCH, and DELETE. These correspond to create, read, update, and delete (or CRUD) operations, respectively. There are a number of other verbs, too, but are utilized less frequently. Of those less-frequent methods, OPTIONS and HEAD are used more often than others.

Below is a table summarizing recommended return values of the primary HTTP methods in combination with the resource URIs:

| HTTP Verb | CRUD | Entire Collection (e.g. /customers) | Specific Item (e.g. /customers/{id}) |
|------------------|----------------|--|--|
| POST | Create | 201 (Created), 'Location' header with link to /customers/{id} containing new ID. | 404 (Not Found), 409 (Conflict) if resource already exists.. |
| GET | Read | 200 (OK), list of customers. Use pagination, sorting and filtering to navigate big lists. | 200 (OK), single customer. 404 (Not Found), if ID not found or invalid. |
| PUT | Update/Replace | 405 (Method Not Allowed), unless you want to update/replace every resource in the entire collection. | 200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid. |
| PATCH | Update/Modify | 405 (Method Not Allowed), unless you want to modify the collection itself. | 200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid. |
| DELETE | Delete | 405 (Method Not Allowed), unless you want to delete the whole collection—not often desirable. | 200 (OK). 404 (Not Found), if ID not found or invalid. |

4.5 Spring Boot

Java and the Spring framework

While Java may be easy to use and easier to learn than other languages, the level of complexity to build, debug, and deploy Java apps has escalated to dizzying new heights. This is due to the exponential number of variables modern developers are faced with when developing web apps or mobile apps for common modern technologies such as music streaming or mobile cash payment apps. A developer writing a basic line-of-business app now needs to deal with multiple libraries, plugins, error logging and handling libraries, integrations with web services, and multiple languages such as C#, Java, HTML, and others. Understandably, there is an

insatiable demand for any tools that will streamline Java app development, saving the developers time and money.

Enter application frameworks—the large bodies of prewritten code that developers can use and add to their own code, as their needs dictate. These frameworks lighten the developer's load for almost any need—whether they're developing mobile and web apps or working with desktops and APIs. Frameworks make creating apps quicker, easier, and more secure by providing reusable code and tools to help tie the different elements of a software development project all together.

Here's where Spring comes in: Spring is an open-source project that provides a streamlined, modular approach for creating apps with Java. The family of Spring projects began in 2003 as a response to the complexities of early Java development and provides support for developing Java apps. The name, Spring, alone usually refers to the application framework itself or the entire group of projects, or modules. Spring Boot is one specific module that is built as an extension of the Spring framework.

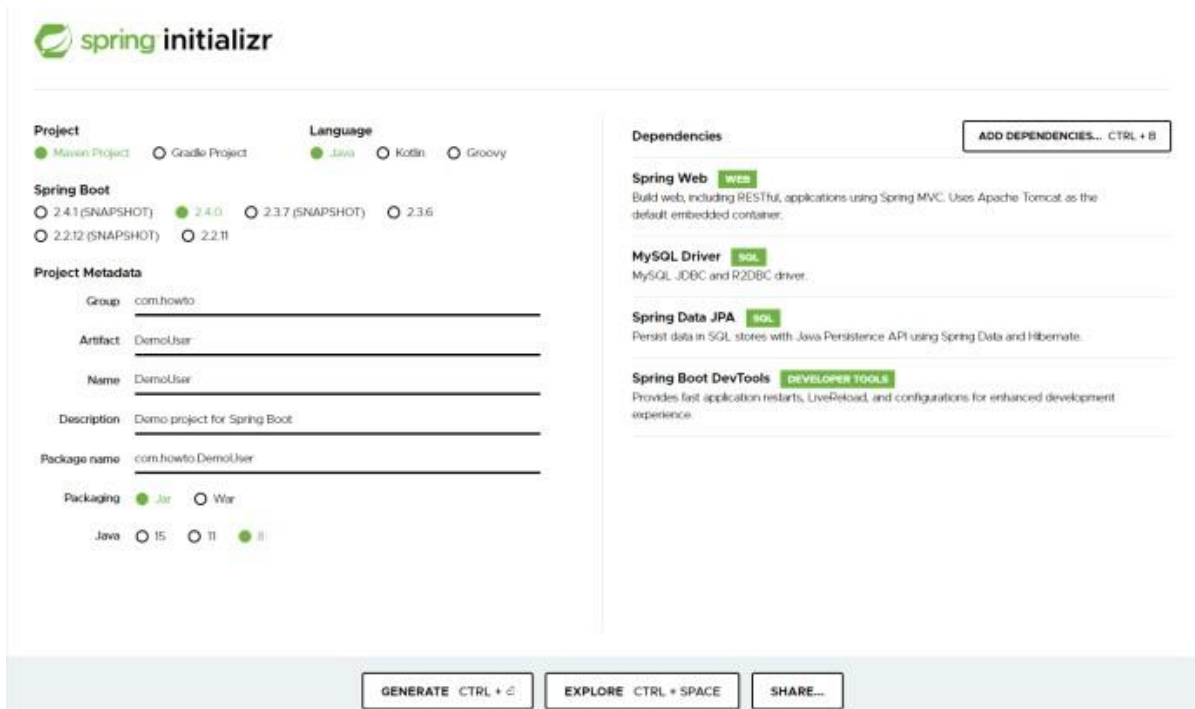
So, with that background on how the Spring framework, Spring Boot, and Java work together, here's the definition of Spring Boot—the tool that streamlines and speeds up web app and microservices development within the Java framework, Spring.

4.5.1 Building a Spring Boot REST API in JAVA

Step 1: Initializing a Spring Boot Project

To start with **Spring Boot REST API**, you first need to initialize the Spring Boot Project. You can easily initialize a new Spring Boot Project with Spring Initializr.

From your Web Browser, go to **start.spring.io**. Choose **Maven** as your Build Tool and Language as **Java**. Select the specific version of Spring Boot you want to go ahead with.

The screenshot shows the Spring Initializr web interface. On the left, under 'Project', 'Maven Project' is selected. Under 'Language', 'Java' is selected. Under 'Spring Boot', version '2.4.0' is selected. The 'Project Metadata' section includes fields for Group (com.howto), Artifact (DemoUser), Name (DemoUser), Description (Demo project for Spring Boot), and Package name (com.howto.DemoUser). The 'Packaging' is set to 'Jar' and the 'Java' version is set to '11'. On the right, the 'Dependencies' section lists 'Spring Web', 'MySQL Driver', 'Spring Data JPA', and 'Spring Boot DevTools'. At the bottom, there are buttons for 'GENERATE', 'EXPLORE', and 'SHARE...'.

Project: ☒ Maven Project ☐ Gradle Project Language: ☒ Java ☐ Kotlin ☐ Groovy

Spring Boot: ☐ 2.4.1 (SNAPSHOT) ☒ 2.4.0 ☐ 2.3.7 (SNAPSHOT) ☐ 2.3.6 ☐ 2.2.12 (SNAPSHOT) ☐ 2.2.11

Project Metadata

Group:

Artifact:

Name:

Description:

Package name:

Packaging: ☒ Jar ☐ War

Java: ☐ 15 ☐ 11 ☒ 8

Dependencies:

Spring Web **WEB**
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

MySQL Driver **SQL**
MySQL JDBC and R2DBC driver.

Spring Data JPA **SQL**
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

Spring Boot DevTools **DEVELOPER TOOLS**
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Fig 4.4: Spring Initializer

You can go ahead and add a few dependencies to be used in this project.

- **Spring Data JPA** – Java Persistence API and Hibernate.
- **Spring Web** – To include Spring MVC and embed Tomcat into your project.
- **Spring Boot DevTools** – Development Tools.
- **MySQL Driver** – JDBC Driver (any DB you want to use).

Once you're done with the configuration, click on "**Generate**". A ZIP file will then be downloaded. Once the download is finished, you can now import your project files as a Maven Project into your IDE (such as **Eclipse**) or a text editor of choice.

Step 2: Connecting Spring Boot to the Database

Next, you need to set up the Database, and you can do it easily with Spring Data JPA.

Add some elementary information in your **application.properties** file to set up the connection to your preferred Database. Add your JDBC connection URL, provide a username and password for authentication, and set the **ddl-auto** property to **update**.

Hibernate has different dialects for different Databases. Hibernate automatically sets the dialect for different Databases, but it's a good practice to specify it explicitly.

```
spring.datasource.url = jdbc:mysql://localhost:3306/user
spring.datasource.username = user
spring.datasource.password = user
spring.jpa.hibernate.ddl-auto = update
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
```

Step 3: Creating a User Model

The next step is to create a **Domain Model**. They are also called **Entities** and are annotated by **@Entity**.

Create a simple User entity by annotating the class with **@Entity**. Use **@Table** annotation to specify the name for your Table. **@Id** annotation is used to annotate a field as the id of an entity. Further, set it as a **@GeneratedValue** and set the **GenerationType** to **AUTO**.

```
@Entity
@Table(name = "user")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;
    private String name;
}
```

Now, this class (entity) is registered with Hibernate.

Step 4: Creating Repository Classes

To perform CRUD (Create, Read, Update, and Delete) operations on the **User** entities, you'll need to have a **UserRepository**. To do this, you'll have to use the **CrudRepository** extension and annotate the interface with **@Repository**.

```
@Repository
```

```
public interface UserRepository extends CrudRepository<User, Long> { }
```

Step 5: Creating a Controller

You've now reached the Business Layer, where you can implement the actual business logic of processing information.

@RestController is a combination of **@Controller** and **@ResponseBody**. Create a **UserController** as shown below.

```
@RestController
@RequestMapping("/api/user")
public class UserController {

    @Autowired
    private UserRepository userRepository;

    @GetMapping
    public List<User> findAllUsers() {
        // Implement
    }

    @GetMapping("/{id}")
    public ResponseEntity<User> findUserById(@PathVariable(value = "id") long id) {
        // Implement
    }

    @PostMapping
    public User saveUser(@Validated @RequestBody User user) {
        // Implement
    }
}
```

You've to **@Autowired** your **UserRepository** for dependency injection. To specify the type of HTTP requests accepted, use the **@GetMapping** and **@PostMapping** annotations.

Let's implement the **findAll()** endpoint. It calls the **userRepository** to **findAll()** users and returns the desired response.

```
@GetMapping
public List<User> findAllUsers() {
    return userRepository.findAll();
}
```

To get each user by their **id**, let's implement another endpoint.

```
@GetMapping("/{id}")
public ResponseEntity<User> findUserById(@PathVariable(value = "id") long id) {
    Optional<User> user = userRepository.findById(id);

    if(user.isPresent()) {
        return ResponseEntity.ok().body(user.get());
    } else {
        return ResponseEntity.notFound().build();
    }
}
```

If the **user.isPresent()**, a **200 OK** HTTP response is returned, else, a **ResponseEntity.notFound()** is returned.

Now, let's create an endpoint to save users. The **save()** method saves a new user if it isn't already existing, else it throws an exception.

```
@PostMapping
public User saveUser(@Validated @RequestBody User user) {
    return userRepository.save(user);
}
```

The **@Validated** annotation is used to enforce basic validity for the data provided about the user. The **@RequestBody** annotation is used to map the body of the **POST** request sent to the endpoint to the **User** instance you'd like to save.

Step 6: Compile, Build and Run

You can change the port of Spring Boot from your **application.properties** file.

```
server.port = 9090
```

8080 is the default port that Spring Boot runs in.

It's time to run the Spring Boot REST API you've created. To run the application, directly execute **./mvnw spring-boot:run** on the command line from your base project folder where **pom.xml** is located.

If your application has successfully run, you'll be able to see these audit logs at the end of your command line.

```
2020-11-05 13:27:05.073 INFO 21796 --- [ restartedMain]
o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
```

```
2020-11-05 13:27:05.108 INFO 21796 --- [ restartedMain]
o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with
context path "
2020-11-05 13:27:05.121 INFO 21796 --- [ restartedMain]
com.howto.DemoUser.DemoUserApplication : Started DemoUserApplication in 1.765
seconds (JVM running for 2.236)
```

Step 7: Testing the Spring Boot REST APIs

Your Spring Boot REST API is now up and running on **http://localhost:8080/**.

Use your browser, **curl**, or **Postman** to test the endpoints.

To send an HTTP GET request, go to **http://localhost:8080/api/user** from your browser and it will display a JSON response as shown.

```
[
  {
    "id": 1,
    "name": "John"
  },
  {
    "id": 2,
    "name": "Jane"
  },
  {
    "id": 3,
    "name": "Juan"
  }
]
```

Now, send an HTTP POST request to add specific users to your Database.

```
$ curl --location --request POST 'http://localhost:8080/api/user'
--header 'Content-Type: application/json'
--data-raw '{ "id": 4, "name": "Jason" }'
```

The API will return a 200 OK HTTP response with the response body of the persisted user.

```
{
  "id": 4,
  "name": "Jason"
}
```

That's it, you have now successfully tested your Spring Boot REST API.

4.6 POSTMAN

Postman is a standalone tool that **exercises web APIs by making HTTP requests from outside the service**. When using Postman, we don't need to write any HTTP client infrastructure code just for the sake of testing. Instead, we create test suites called collections and let Postman interact with our API.

4.6.1 Testing API's using Postman

Adding API tests

You can connect a test collection (a collection containing API tests) to an API you've defined in the Postman API Builder.

To add a test collection to an API, do the following:

1. Select **APIs** in the sidebar and select an API.
2. Select **Test and Automation**.
3. Next to **Collections**, select + and select an option:
 - **Add new collection** - This option creates a new empty collection in the API. You can add your tests to the **Tests** tab.
 - **Copy existing collection** - Select an available collection from the list. A copy of the collection is added to the API.
 - **Generate from definition** - Change any settings to customize the new collection and select **Generate Collection**.

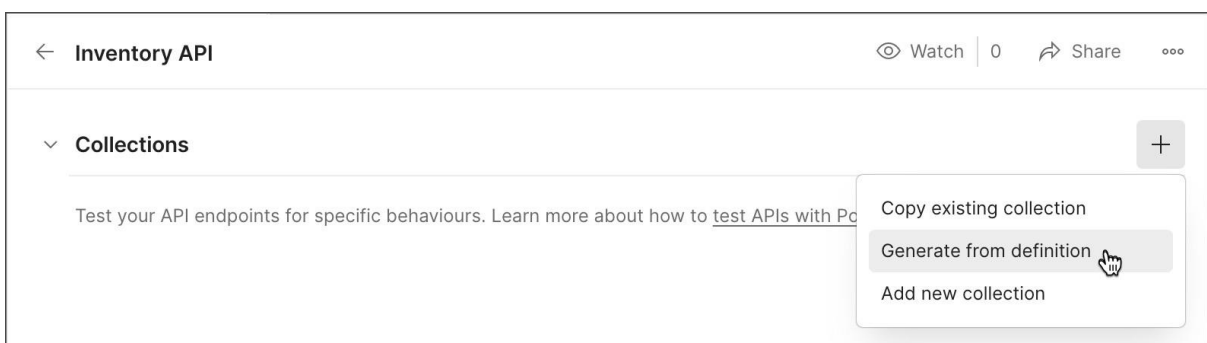


Fig 4.5: Postman adding tests

For more information on how to write API tests, see [Writing tests](#).

Running API tests

After adding a test collection, you can run the collection to test your API and view test results.

To run a test collection for an API, do the following:


1. Select **APIs** in the sidebar and select an API.
2. Select **Test and Automation**.
3. Under **Collections**, select  **Run** next to a test collection.
4. Select any configuration options for the collection run, then select **Run API Tests**.
Learn more about using the Collection Runner.
5. To view detailed test results, expand the collection and select **View Report** next to a test run.



Fig 4.6: Postman adding collection

To remove a test collection from an API, select the delete icon  next to the collection.

4.7 PROJECT

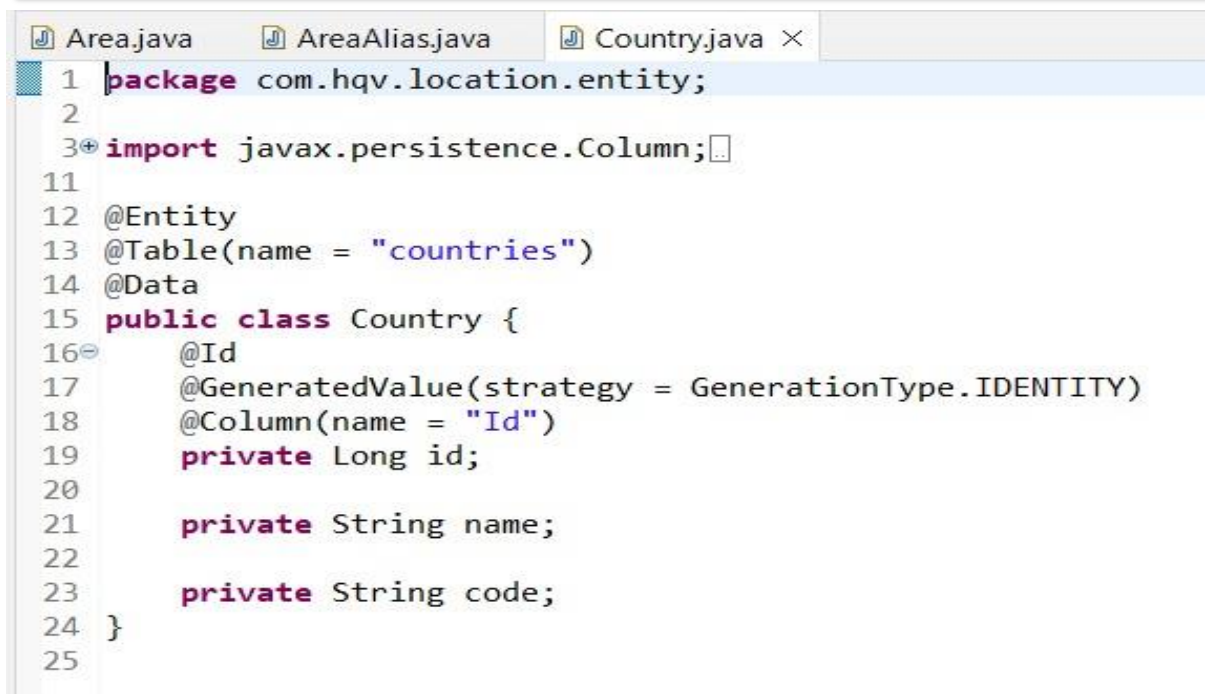
Location API's Using Java Spring Boot

```

1 package com.hqv.location.pojo;
2
3 import lombok.Data;
4
5 @Data
6 public class CountryDto {
7     private Long id;
8     private String name;
9     private String code;
10
11 }
12
13
14
15

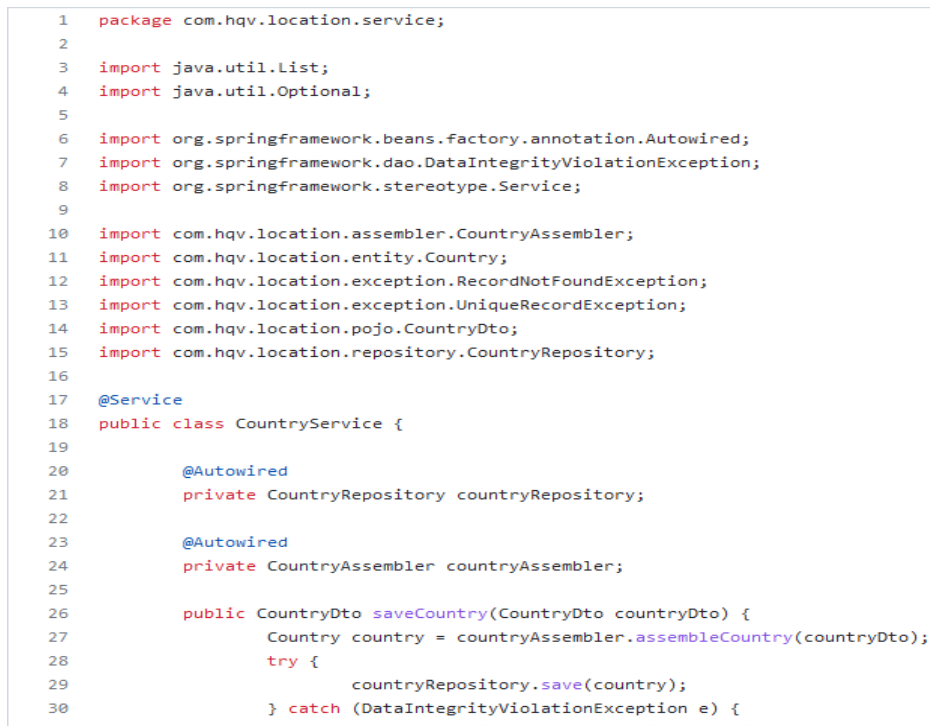
```

Fig 4.7: Country Dto class



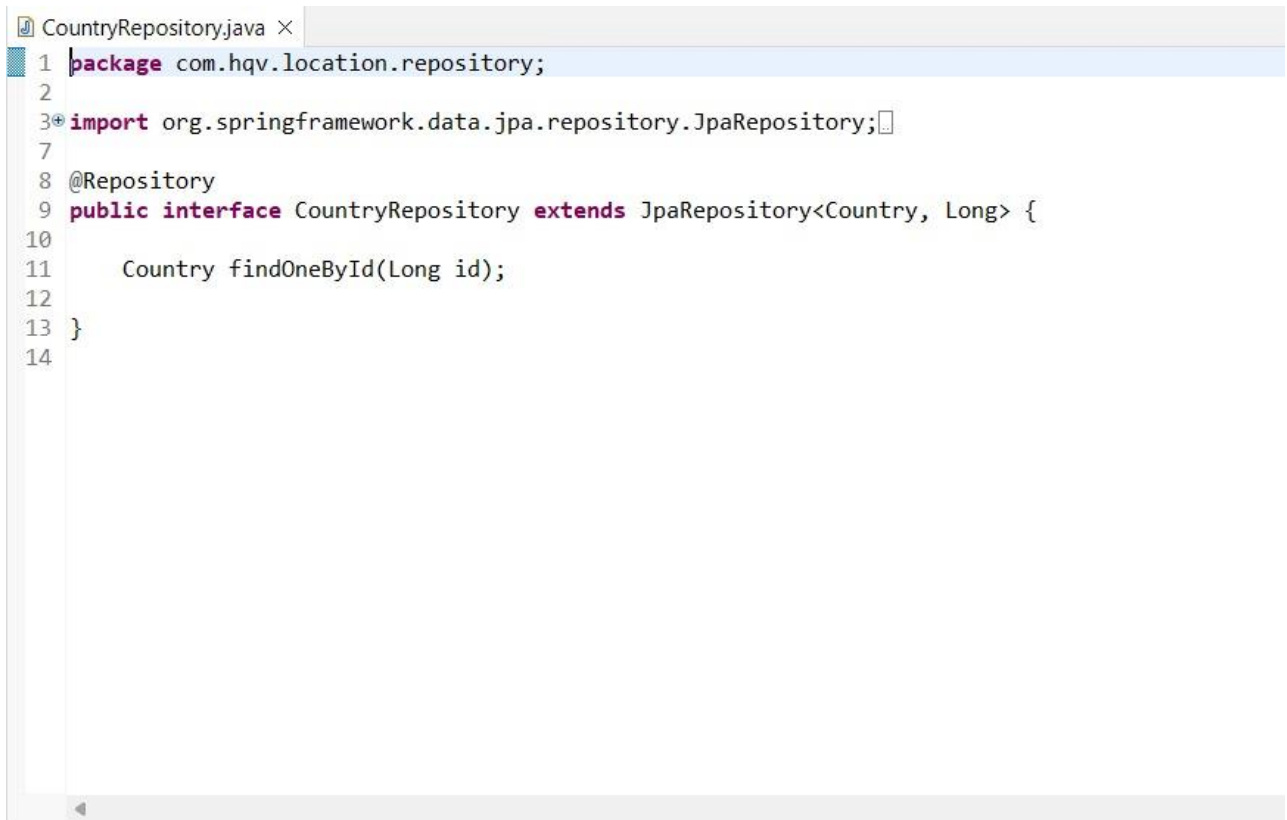
```
1 package com.hqv.location.entity;
2
3 import javax.persistence.Column;
4
11
12 @Entity
13 @Table(name = "countries")
14 @Data
15 public class Country {
16     @Id
17     @GeneratedValue(strategy = GenerationType.IDENTITY)
18     @Column(name = "Id")
19     private Long id;
20
21     private String name;
22
23     private String code;
24 }
25
```

Fig 4.8: Country Entity class



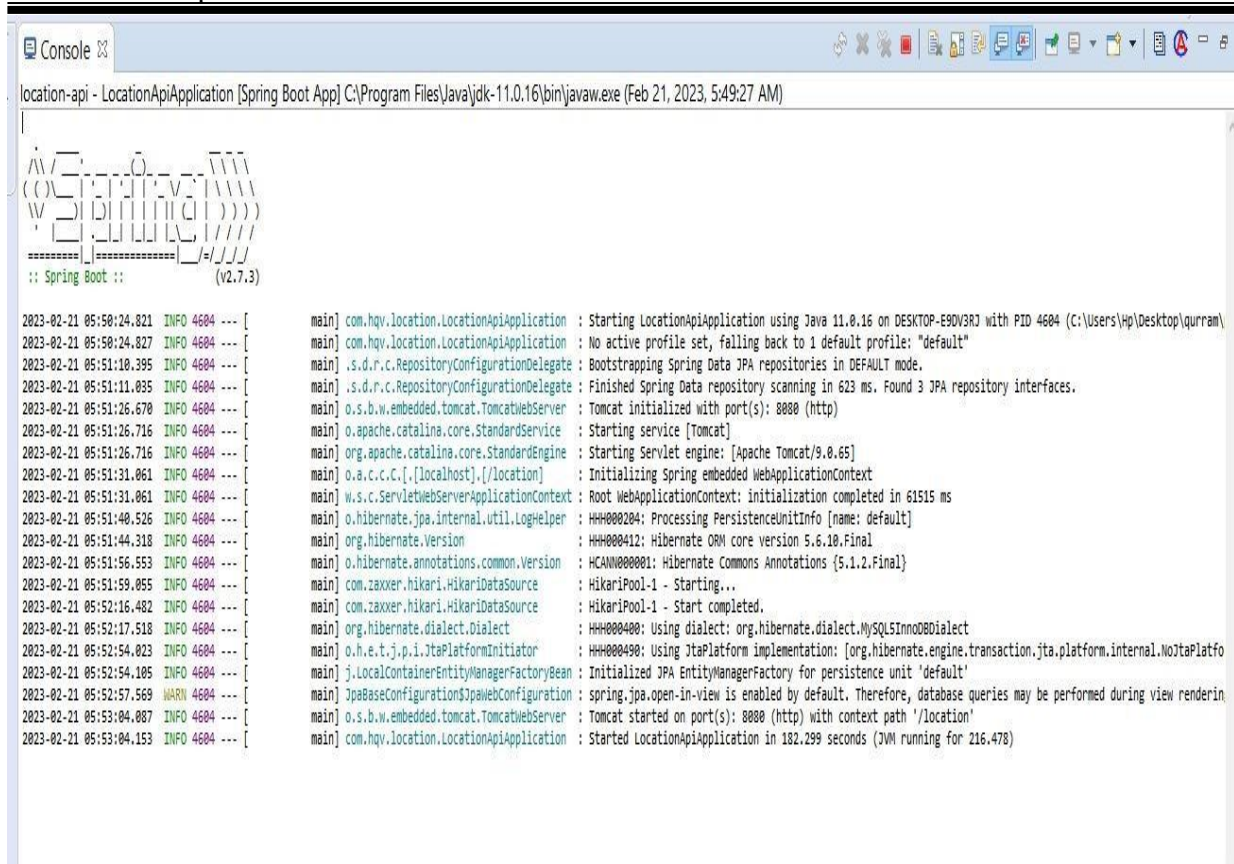
```
1 package com.hqv.location.service;
2
3 import java.util.List;
4 import java.util.Optional;
5
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.dao.DataIntegrityViolationException;
8 import org.springframework.stereotype.Service;
9
10 import com.hqv.location.assembler.CountryAssembler;
11 import com.hqv.location.entity.Country;
12 import com.hqv.location.exception.RecordNotFoundException;
13 import com.hqv.location.exception.UniqueRecordException;
14 import com.hqv.location.pojo.CountryDto;
15 import com.hqv.location.repository.CountryRepository;
16
17 @Service
18 public class CountryService {
19
20     @Autowired
21     private CountryRepository countryRepository;
22
23     @Autowired
24     private CountryAssembler countryAssembler;
25
26     public CountryDto saveCountry(CountryDto countryDto) {
27         Country country = countryAssembler.assembleCountry(countryDto);
28         try {
29             countryRepository.save(country);
30         } catch (DataIntegrityViolationException e) {
31
32         }
33     }
34 }
```

Fig 4.9: Country Service class



```
CountryRepository.java ×
1 package com.hqv.location.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5
6
7
8 @Repository
9 public interface CountryRepository extends JpaRepository<Country, Long> {
10
11     Country findOneById(Long id);
12
13 }
14
```

Fig 4.11 Country Repository class



```

location-api - LocationApiApplication (Spring Boot App) C:\Program Files\Java\jdk-11.0.16\bin\javaw.exe (Feb 21, 2023, 5:49:27 AM)

:: Spring Boot ::
(v2.7.3)

2023-02-21 05:50:24.821 INFO 4604 --- [main] com.hqy.location.LocationApiApplication : Starting LocationApiApplication using Java 11.0.16 on DESKTOP-E90V3R3 with PID 4604 (C:\Users\hp\Desktop\qurran\
2023-02-21 05:50:24.827 INFO 4604 --- [main] com.hqy.location.LocationApiApplication : No active profile set, falling back to 1 default profile: "default"
2023-02-21 05:51:10.395 INFO 4604 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2023-02-21 05:51:11.035 INFO 4604 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 623 ms. Found 3 JPA repository interfaces.
2023-02-21 05:51:26.670 INFO 4604 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2023-02-21 05:51:26.716 INFO 4604 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-02-21 05:51:26.716 INFO 4604 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.65]
2023-02-21 05:51:31.061 INFO 4604 --- [main] o.a.c.c.C.[.localhost.]/location : Initializing Spring embedded WebApplicationContext
2023-02-21 05:51:31.061 INFO 4604 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 61515 ms
2023-02-21 05:51:40.526 INFO 4604 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2023-02-21 05:51:44.318 INFO 4604 --- [main] org.hibernate.Version : HHH000412: Hibernate ORM core version 5.6.10.Final
2023-02-21 05:51:56.553 INFO 4604 --- [main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
2023-02-21 05:51:59.055 INFO 4604 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2023-02-21 05:52:16.482 INFO 4604 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2023-02-21 05:52:17.518 INFO 4604 --- [main] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.MySQL5InnoDBDialect
2023-02-21 05:52:54.023 INFO 4604 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2023-02-21 05:52:54.105 INFO 4604 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2023-02-21 05:52:57.569 WARN 4604 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view renderin
2023-02-21 05:53:04.087 INFO 4604 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path '/location'
2023-02-21 05:53:04.153 INFO 4604 --- [main] com.hqy.location.LocationApiApplication : Started LocationApiApplication in 182.299 seconds (JVM running for 216.478)

```

Fig 4.12: Console Output

GET localhost:8080/location/country Send

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies

Query Params

| KEY | VALUE | DESCRIPTION | ... | Bulk Edit |
|-----|-------|-------------|-----|-----------|
|-----|-------|-------------|-----|-----------|

Body 200 OK 26 ms 427 B Save Response

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 1,
4     "name": "Afganistam",
5     "code": "AF"
6   },
7   {
8     "id": 2,
9     "name": "Aland Islands",
10    "code": "AX"
11  },
12 ]
```

Cookies Capture requests Runner Trash

Fig 4.13: Country HTTP Response

CHAPTER

REFLECTION

The internship from HQV Technologies was very useful. The instruction was useful and showed us how to quickly construct solutions to the issue statements. The most advantageous aspect of internships was the opportunity to get practical knowledge on different technologies. The instructor used a distinct method of instruction, giving us time to apply the principles realistically to ensure that we understood them. We were able to successfully complete the exam because to the effective teaching we received throughout the internship.

The idea of internship program sees merit in attempting to shorten the period on training that is often significant duration to orient the trainee or newly inducted person onto the project. The internship covered the concepts of back end development object oriented concepts, working with GitHub , Understanding, developing and testing API's, and also some UI/UXdesign such as UI Mockup. It is and has great tools, libraries and frameworks.

The internship session has been a great learning journey helping the participants in the internship program to understand the concepts of Web Development. It also helped us improve our logical thinking. It helped us to improve our communication skills. They taught us to manage the time so that we could code maximum in limited or specified time. We realized that soft skills contribute to a positive work environment and help us maintain an efficient workflow.

CHAPTER

CONCLUSION

The demand for internships is more pressing as self-learning skills are increasingly needed in the workplace on short notice. I learned about industry norms, the abilities to study on our own, logical reasoning, and many other skills that are needed to help us contribute to and grow with the industry. I now have a better understanding of the professional options available thanks to the internship program. It becomes clearer how important it is to continue learning throughout one's career in order to succeed in the field, as well as how important it is to approach challenges with an open mind.

Interpersonal skills were enhanced by the internship. During this internship, we developed our web page design skills. Our logical reasoning skills were also improved by the trainer.