Zopstore Web App

Project report submitted in partial fulfillment for the requirement for the degree of Bachelor of Technology

In

Computer Science and Engineering/Information Technology

By

Piyush Singh (191276)

Under the supervision of

Prof. (Dr.) Vivek Kumar Sehgal

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

CERTIFICATE

I hereby declare that the work presented in this report entitled **"Zopstore Web App"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from July 2022 to May 2023 under the supervision of **Prof Dr Vivek Kumar Sehgal, Professor and Head, Fellow IEI, SM-IEEE, SM-ACM,** Department of CSE Jaypee University of Information Technology, Waknaghat.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Piyush Singh

Piyush Singh, 191276.

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Prof (Dr) Vivek Kumar Sehgal
Professor and Head, Fellow IEI, SM-IEEE, SM-ACM
Computer Science and Engineering and Information Technology
Dated:

PLAGIARISM CERTIFICATE

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: .....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: _____ Department: _____ Enrolment No _____

Contact No. _____ E-mail. _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

Piyush Singh

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at .....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

.....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to Lord Shiva for His divine blessing makes it possible for us to complete the project work successfully.

I am grateful and wish my profound indebtedness to **Prof Dr Vivek Kumar Sehgal, Professor and Head, Fellow IEI, SM-IEEE, SM-ACM,** Department of CSE, for his keen interest in the field of "**Machine Learning**" to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project. I would like to express my heartiest gratitude for his kind help to finish my project.

I would also generously welcome my friend Suveer Sharma and each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

No expression of appreciation is complete without recognition of the prayers, good wishes, advice and moral support of my affectionate parents, which helped me immensely to achieve my goal.

Name: Piyush Singh

Enrollment No.: 191276

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Making a web application is very simple, but testing, organising, cleaning up, and maintaining the code is difficult. We adhere to the Three-Layered Architecture and the Go programming language to address this.

The handler, service, and datastore layers are independent of one another. The handler layer parses the request body after receiving it to extract any pertinent data. After calling the service layer, where the program's whole logic is specified, the response is subsequently written to the response writer. This layer also communicates with the datastore layer. After obtaining what it needs from the handler layer, it invokes the datastore layer. The datastore layer holds all of the data.

CHAPTER 01:
INTRODUCTION

1.1 Company Profile

A leading provider of retail technology, ZopSmart Technology offers all the resources needed to launch an online store. We offer services including digital marketing, m-commerce, e-wallets, etc.

We assist clients in defining business processes and determining the best integration requirements by utilising sector best practises to increase automation for data communication and better decision making, allowing businesspeople to concentrate on their core competencies while technology handles data communication and facilitates better decision making.

Figure 1: ZopSmart Technologies

Services:

Website Development: Complete website development, from UI/UX design through website development or client website maintenance.
Framework Development: Framework Development for Java, Golang, and NodeJS Internal/Client Webservices.
Website for a company that develops mobile applications: https://zopsmart.com

1.2 Introduction

This programme is designed for store owners who have access to a database of data on online goods. Our goal is to make it simple for executives and administrators to automate their tasks with minimal effort. For instance, they can quickly produce quotes for any inquiries about online products.

1.3 Problem Statement and Motivation for project:

As online business has grown quite prevalent, regardless of age or gender, it is now simple to locate online web retailers on the internet.
When pleasing Customers and Employees of the respective online shopping, using manual techniques may open the door for a number of challenges. These manual transactions squander the Owners' and Employees' valuable time and money. These obstacles directly impact the dealership's profits, owners' interests, and both.

1.4 Objective:

The following are the project's goals:

- Eliminate paper-based labour from the shopping area, such as using diaries to record brand and product information.

- Keeping Product and Brand details in many locations (such as a diary or mobile device) and by multiple people (the owner and employees) will eliminate data redundancy.

- Stop wasting the Owner's and Employees' time, resources, efforts, and money.

- Enhance the effectiveness and efficiency of a web store's operations, services, and procedures including storing Product and Brand information.

- Boost employee and owner satisfaction.

1.5 Purpose and Methodology:

The "Product-Brand Store Web API" project is a Web API. Complete details about the goods and its brand are provided through these API. The foundation of the online Store is a product database that houses all of the necessary product data. It offers the ability to locate, create, update, and delete.
important details regarding the project:

- allowing for the addition of Product and Brand details.
- enabling the updating of Product and Brand details.
- enabling the display of information about all products and brands.

CHAPTER 02:

LITERATURE SURVEY

2.1 Technology and Literature Review:

2.1.1 Development Tools:

Goland

An integrated development environment (IDE) used in computer programming specifically for the Go language is called Goland Community Edition.JetBrains, a Czech firm, created it. A free, feature-rich, and extendable IDE for independent developers, open-source projects, instruction, and scholarly investigation is called Goland Community version. Goland works on Windows, MacOS, and Linux.

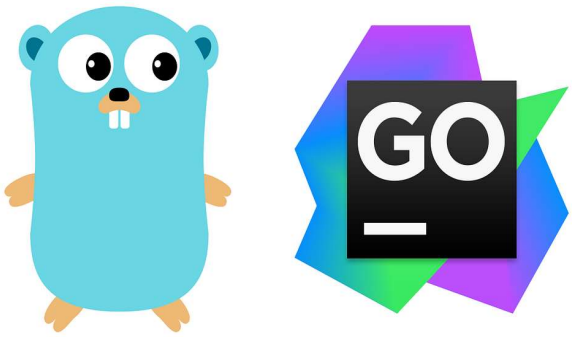

Figure 2: Golang

Language: - Go Language

Go is a general-purpose open source programming language that is sometimes known as Golang or Go.To construct stable and effective software, Google engineers created the language Go.

Go is statically typed and explicit, most closely resembling C. A

compiled language is Go. Go is utilised for many different applications, including DevOps, Write Rest Api, cloud and server-side applications, and more.

Figure 3: Golang

Database: - MySQL

The core of Oracle's relational database management system (RDBMS), MySQL, is the structured query language (SQL).

A database is a planned collection of data. It might be anything from a simple grocery list to a photo gallery or a place to store the massive volumes of data in a corporate network. In particular, a relational database is a digital repository that gathers data and organizes it according to the relational paradigm. In this paradigm, tables are made up of rows and columns, and relationships between data items all follow a strict logical structure. An RDBMS is a collection of software tools used to set up, run, and query such a database.

Figure 4: MySQL

Postman

design, construction, and documentation of APIs. Developers may send queries, view responses, and analyse data from APIs thanks to the user-friendly interface it offers. Developers may manage and run tests for various endpoints and scenarios by creating and organising groups of API calls in Postman.

One of Postman's standout features is its support for a variety of HTTP methods, including GET, POST, PUT, DELETE, and more. This capability enables developers to interact with APIs and test various activities. Additionally, it supports many forms of authentication, including OAuth and API keys, allowing developers to test and validate secure API endpoints.

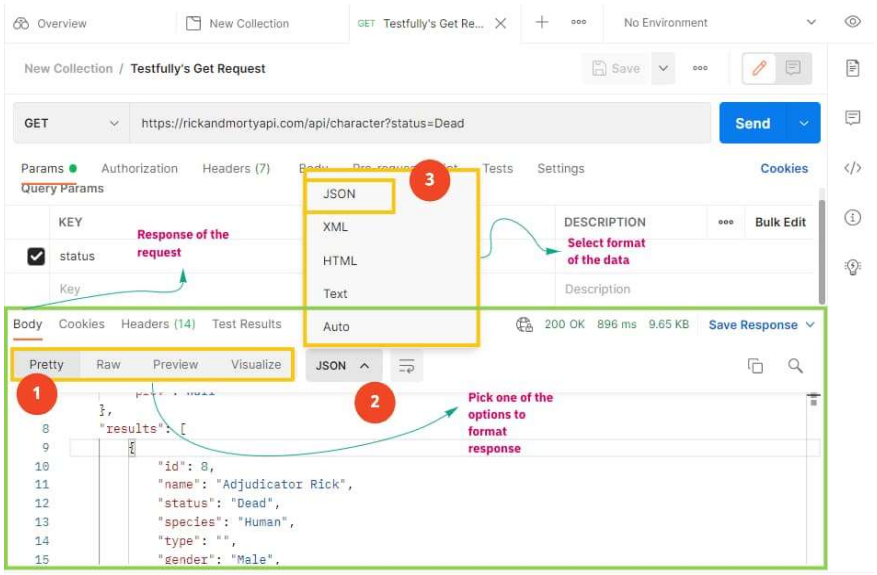

Figure 5: Postman

A wide range of testing options are provided by Postman, including the ability to create and execute JavaScript-based automated tests. As a result, programmers can specify assertions and validations to make sure that API replies satisfy predetermined requirements. It is convenient to carry out in-depth testing and confirm the behaviour of APIs by executing test scripts as a component of collections or individual requests.

Postman also makes it easier to document APIs by offering tools for creating interactive and shareable documentation. For the sake of documentation, developers can include explanations, illustrations, and even mock servers that replicate API responses. Teams may work together and inform stakeholders about API requirements more easily as a result.

Version Control: - Git/GitHub

GitHub is a web-based platform for version control and collaboration for software engineers. Microsoft, GitHub's largest individual donor, bought the service for $7.5 billion in 2018. Using a software as a service (SaaS)

delivery model, GitHub was founded in 2008. Its foundation was Git, an open-source code management system created by Linus Torvalds to hasten software development.

Git is a tool for storing project source code and tracking all code alterations. It enables developers to work on a project more successfully by offering methods for addressing possibly conflicting modifications from different developers.

GitHub's public repositories allow for the free modification, adaptation, and improvement of software by developers; nevertheless, the firm provides a number of paid plans for private repositories. Both public and private repositories hold all of a project's files together with each file's revision history. Repositories can have several collaborators and can be either public or private.

Figure 6: Git and GitHub

Container Manager: - Docker

Developers may automate the deployment and administration of applications inside compact, portable containers using the open-source technology known as Docker. Applications and their dependencies are packaged inside self-contained Docker containers, which enable them to function consistently in a variety of contexts.

An overview of Docker and Docker containers is provided here.

Containerization: Docker makes use of containerization technology, which enables the packaging of applications into independent, portable containers. Each container contains the configuration files, dependencies, and libraries needed for the application to function. Application behaviour is guaranteed to be consistent and reproducible across the underlying infrastructure thanks to container environments.

Docker containers are extremely portable. They make it simple to deploy and operate the same container on various devices, operating systems, or cloud platforms since they package the programme and its dependencies into a single package. This portability solves the "it works on my machine" issue and makes switching between the development, testing, and production environments for apps much easier.

Resource Efficiency: Docker containers are effective users of resources. Docker shares the operating system kernel of the host machine, in contrast to traditional virtualization, which runs several virtual machines with their own operating systems. Compared to virtual machines, this method lowers overhead and maximises resource use, enabling the use of multiple containers on a single host machine.

Isolation: Docker containers offer some level of separation between the host system and the applications. Applications and the dependencies on them are kept separate from one another by operating in separate, isolated environments in each container. By avoiding conflicts and giving users better control over the runtime environment, this isolation improves security, stability, and dependability.

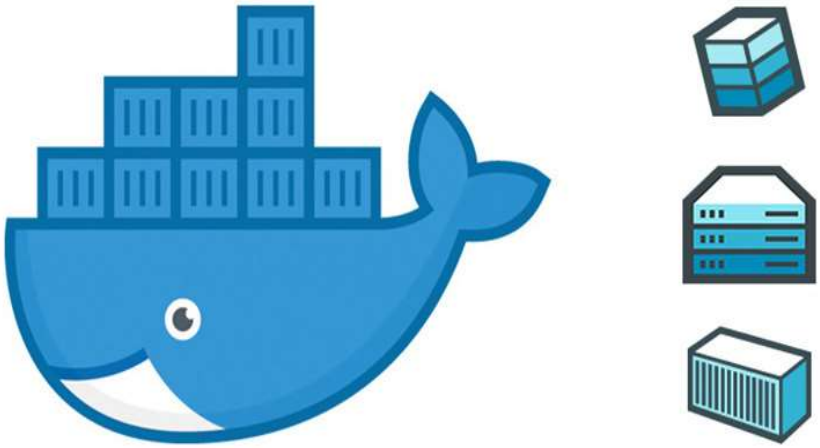

Figure 7: Docker

Container Orchestration - Kubernetes

The deployment, scaling, and maintenance of containerized applications are all automated via the open-source container orchestration technology known as Kubernetes (also known as K8s). The Cloud Native Computing Foundation (CNCF) now maintains it after Google initially built it. In a distributed context, Kubernetes offers a reliable and scalable method for managing containerized workloads.

Kubernetes is made specifically for managing and orchestrating containers. It enables you to specify and deploy application workloads as a group of containers collectively referred to as pods, which serve as the fundamental scheduling units. The placement, scalability, and lifetime management of these pods are handled by Kubernetes, ensuring that the application is kept in the appropriate condition.

High availability and scalability: Kubernetes makes it simple to scale applications. To accommodate more traffic or demand, the number of pods can be horizontally scaled. Assuring high availability and balancing the load, Kubernetes automatically distributes the pods among the

available nodes. Additionally, it keeps an eye on the health of the pods and restarts or reschedules them when there are errors or node disturbances.

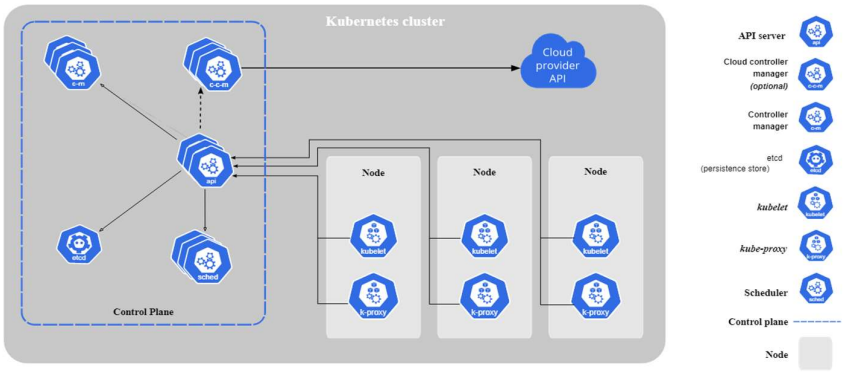

Figure 8: Kubernetes

Terraform And Terragrunt

Two well-known open-source technologies, Terraform and Terragrunt, make declarative and scalable infrastructure provisioning and management possible. While Terraform is primarily concerned with orchestrating infrastructure, Terragrunt expands its functionality with new tools to improve code reuse and maintainability.

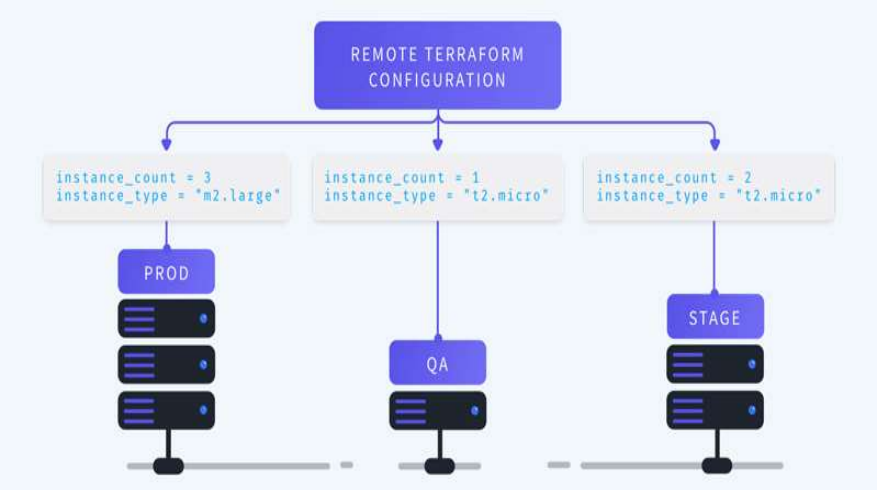

Figure 9: Terraform and Terragrunt

Terraform:

HashiCorp created the infrastructure-as-code (IaC) tool known as Terraform. You may design and manage infrastructure resources across different cloud service providers, data centres, and service providers thanks to this. Utilising a high-level configuration language, typically written in HashiCorp Configuration Language (HCL), Terraform allows you to describe your infrastructure. Among Terraform's most important attributes are:

- Declarative Infrastructure: Terraform enables you to specify in configuration files the desired state of your infrastructure. It offers a declarative approach to infrastructure management and automatically manages the provisioning and management of resources to fit that desired state.

- Support for Multiple Cloud Providers: Terraform offers support for numerous cloud service providers, including Google Cloud Platform (GCP), Microsoft Azure, Amazon Web Services (AWS), and many others. This enables you to manage resources using a single configuration across several cloud environments.

- Resource Graph and Dependency Management: Based on your configuration, terraform creates a resource dependency graph that aids in its comprehension of the links between resources. This makes it possible for Terraform to effectively plan and carry out infrastructure modifications while retaining the necessary dependencies.

- Infrastructure as Code: Terraform considers infrastructure as code, which enables you to version-control, review, and publish your infrastructure configuration just like any other piece of

code. Collaboration, reproducibility, and best practises in infrastructure management are encouraged by this.

Terragrunt

- Terragrunt is a lightweight configuration layer and wrapper for Terraform, which was also created by HashiCorp. It adds further capabilities on top of Terraform to improve configuration management, code reuse, and modularization. Among Terragrunt's prominent characteristics are:

- Avoid Repetition by Using DRY Configuration: You may abstract and reuse popular Terraform setups with Terragrunt. With its support for configuration inheritance and composition, you may create reusable modules and maintain the DRY principle in your infrastructure code.

- Management of Remote States: Terragrunt makes Terraform's management of remote states simpler. It offers integrated support for remote state storage and enables state sharing between different Terraform deployments. By doing this, team members can collaborate more easily and state management is ensured to be consistent.

- Environment management: By abstracting common configuration patterns, Terragrunt aids in the management of many environments (such as development, staging, and production). It enables you to dynamically apply configurations tailored to certain circumstances, minimising duplication and fostering consistency between contexts.

- Terragrunt has functionality for verifying and testing Terraform setups. Configuration validation. You can use it to run tests against the infrastructure code, verify module setups, and

perform pre-flight checks.

- Management of Dependencies: Terragrunt assists in the management of dependencies among Terraform modules. It makes sure that the modules in your infrastructure codebase are initialised, versioned, and utilised correctly. This makes module management easier and guarantees uniform module usage across all projects.

It is simpler to define, provide, and manage infrastructure resources in a consistent and scalable manner when using Terraform and Terragrunt, two potent technologies that enable infrastructure-as-code practises. They offer effective techniques to automate infrastructure provisioning, encourage teamwork, and spread the word about good infrastructure management practises.

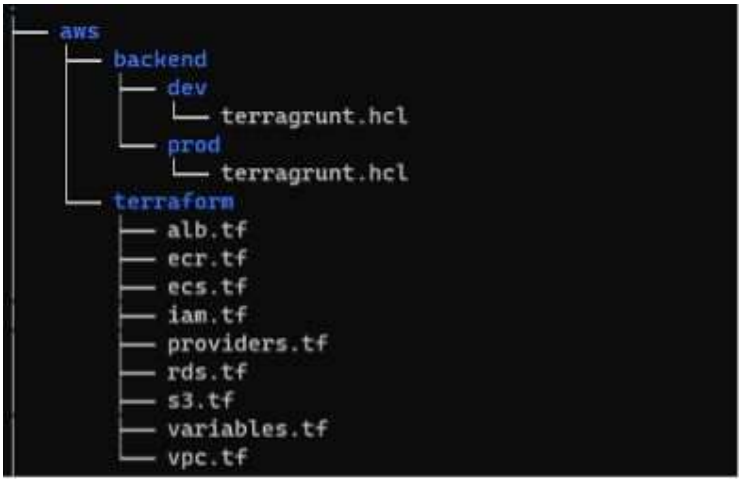

Figure 10: Terragrunt Infra

Helm

Helm is a well-liked open-source package manager for Kubernetes that makes it easier to deploy and manage services and applications. With only one command, users can define, install, and upgrade sophisticated Kubernetes apps. Application configurations may be shared and reused

more easily thanks to Helm's standardised approach to packaging, distributing, and managing Kubernetes manifests.

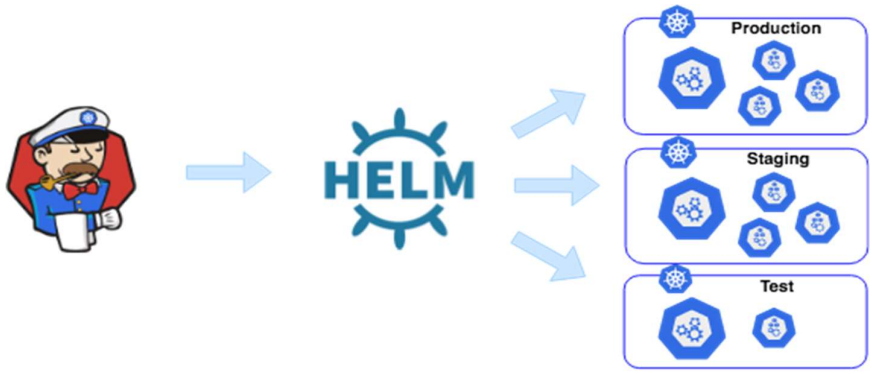

Figure 11: Helm

Swagger

Used for creating, constructing, consuming, and documenting RESTful APIs. It offers a uniform language for describing API endpoints, their input and output parameters, required levels of authentication, and more. Tools and frameworks can use the machine-readable format provided by Swagger to create interactive documentation, client SDKs, and server stubs.

The ability of Swagger to produce interactive, human-readable API documentation is one of its main advantages. Developers may automatically create documentation that include information on endpoints, their supported HTTP methods, request and response types, and even sample payloads by specifying API specifications in a Swagger-compliant manner. Developers may better understand and use APIs thanks to this documentation, which speeds up development and improves integration.

Additionally, Swagger encourages uniformity and standardisation in API design. Swagger makes guarantee that APIs meet best practises and a set of standards by giving a standardised and unambiguous approach to specify API contracts. The usefulness and maintainability of APIs are

enhanced by this uniformity, making it simpler for developers to interact with and expand on already-existing services.

The automated generation of client and server code is another feature of Swagger. Developers can produce client SDKs in a variety of programming languages by utilising the Swagger specification, which will save them time and effort when connecting with APIs. Swagger can be used to generate boilerplate code or server stubs that implement the API contracts on the server side, which speeds up the creation of API backends.

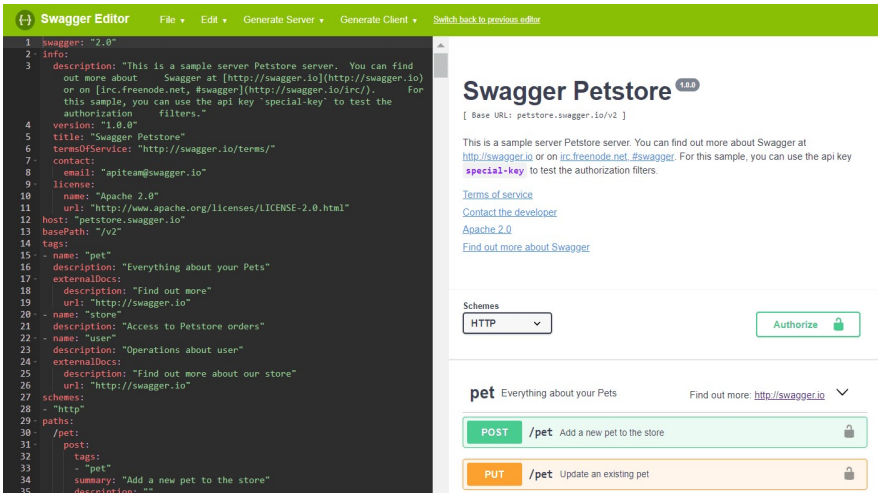

Figure 12: Swagger

2.1.2 Representational State Transfer (REST)

The architectural design approach known as REST (Representational State Transfer) is used to create networked applications. It offers a collection of guidelines and limitations for creating scalable, stateless, and cooperative web services. Application Programming Interfaces (APIs) that follow the REST principles are often utilised.

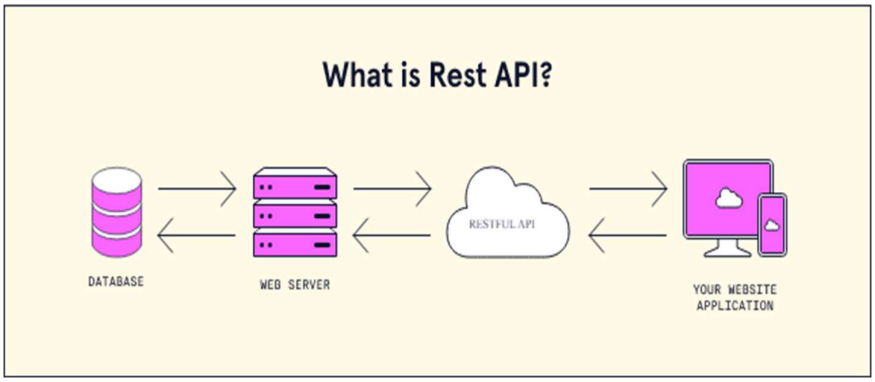

Figure 13: Rest API

Here are the key aspects of REST:

- Stateless Communication: In REST, each request made by a client to a server is complete with all the details the server needs to comprehend and handle it. Between requests, the server doesn't keep any client-specific state. Statelessness improves scalability and makes server-side implementation less complicated.

- Resources are treated as the basic building elements of an application in resource-oriented design, or REST. Any identifiable entity, including data objects, services, and even groups of other resources, can be considered a resource. Each resource has a distinct URI (Uniform Resource Identifier), also known as a URL (Uniform Resource Locator).

- Uniform Interface: To encourage simplicity and interoperability, REST places a strong emphasis on a uniform interface. It specifies a collection of commonly used HTTP methods for altering resources, including GET, POST, PUT, and DELETE. These techniques are used to carry out tasks such as resource retrieval, creation, updating, and deletion.

- Resources are represented in REST using formats like JSON (JavaScript Object Notation) or XML (eXtensible Markup

Language). The representation format can be agreed upon by the client and server through content negotiation, allowing for the most efficient data exchange.

- REST isolates the client (user interface or application) from the server (data storage or service provider) and uses stateless client-server communication. The client can interact with resources via a set of clearly defined APIs that the server exposes. Due to its independence, the client and server can develop separately and with loose connection.

- **HATEOAS**: Hypermedia as the Engine of Application State An essential REST concept known as HATEOAS promotes self-descriptive APIs. This principle states that each server response contains a link to relevant information or activities the client can do. This makes it possible for the client to dynamically learn about and browse the capabilities of the application.

- Layered architecture and caching: REST provides caching to enhance performance and lessen server load. Based on the cacheability guidelines that the server provides, clients can cache answers. In order to offer scalability, security, or other features, intermediaries (proxies, gateways) can be inserted between the client and server according to REST's support for layered architecture.

RESTful APIs' simplicity, scalability, and compatibility have led to their extensive usage. They are frequently employed in the development of distributed systems, microservices, and web services. The foundation for developing scalable, loosely linked systems that may be used by a variety of clients, such as web browsers, mobile devices, and other apps, is provided by RESTful principles.

2.1.3 Golang HTTP Package

An HTTP package is available in the Go programming language's (Golang) comprehensive standard library. Go's HTTP package offers tools for creating HTTP servers, sending HTTP requests, managing sessions, handling cookies, and more. It makes web application development easier and makes it simple for developers to communicate with HTTP-based APIs. The Go HTTP package's main attributes and capabilities are shown below:

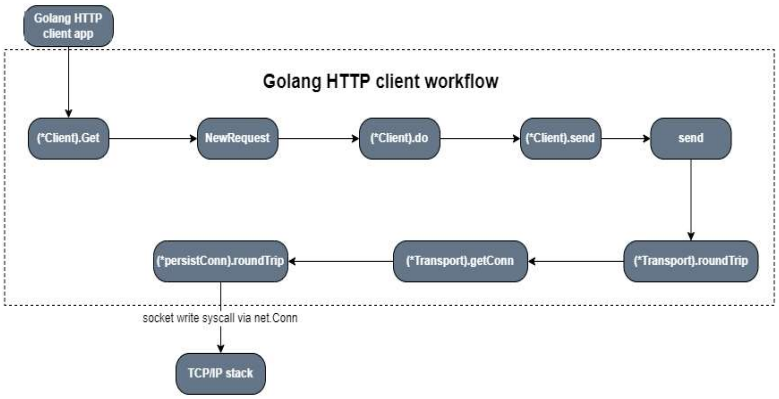

Figure 14: Golang HTTP client workflow

Http Server

You may build HTTP servers and manage incoming requests using the http package.

It offers utilities to register handlers for various HTTP methods and URL paths, enabling you to create your own unique logic for managing particular routes.

The programme offers support for managing custom error pages, redirection, and handling static files.

The server can be set up to handle TLS/SSL encryption, listen on a certain port, and have timeouts that you specify.

The HTTP Client

An HTTP client is included in the http package, which enables you to send HTTP requests to remote servers.

Making GET, POST, PUT, DELETE, and other HTTP requests is supported.

You can control response headers and bodies as well as specify request headers and pass query parameters.

The package offers tools for controlling cookies, following redirection, and establishing request timeouts.

Using the encoding/json package, it also supports sending and receiving JSON data.

Middleware

Middleware can be used to manage routine chores and implement cross-cutting issues thanks to Go's HTTP package.

HTTP handlers can add authentication, logging, rate limitation, and any other required behaviour using middleware functions.

A HandlerFunc type is offered by the http package and enables the chaining of several middleware functions.

2.1.4 Three Layered architecture

A software design pattern called three-layered architecture, or three-tier architecture, divides an application's components into three different layers: the display layer, the business logic layer, and the data storage layer. By requiring a distinct separation of interests, this architectural design encourages modularity, scalability, and maintainability.

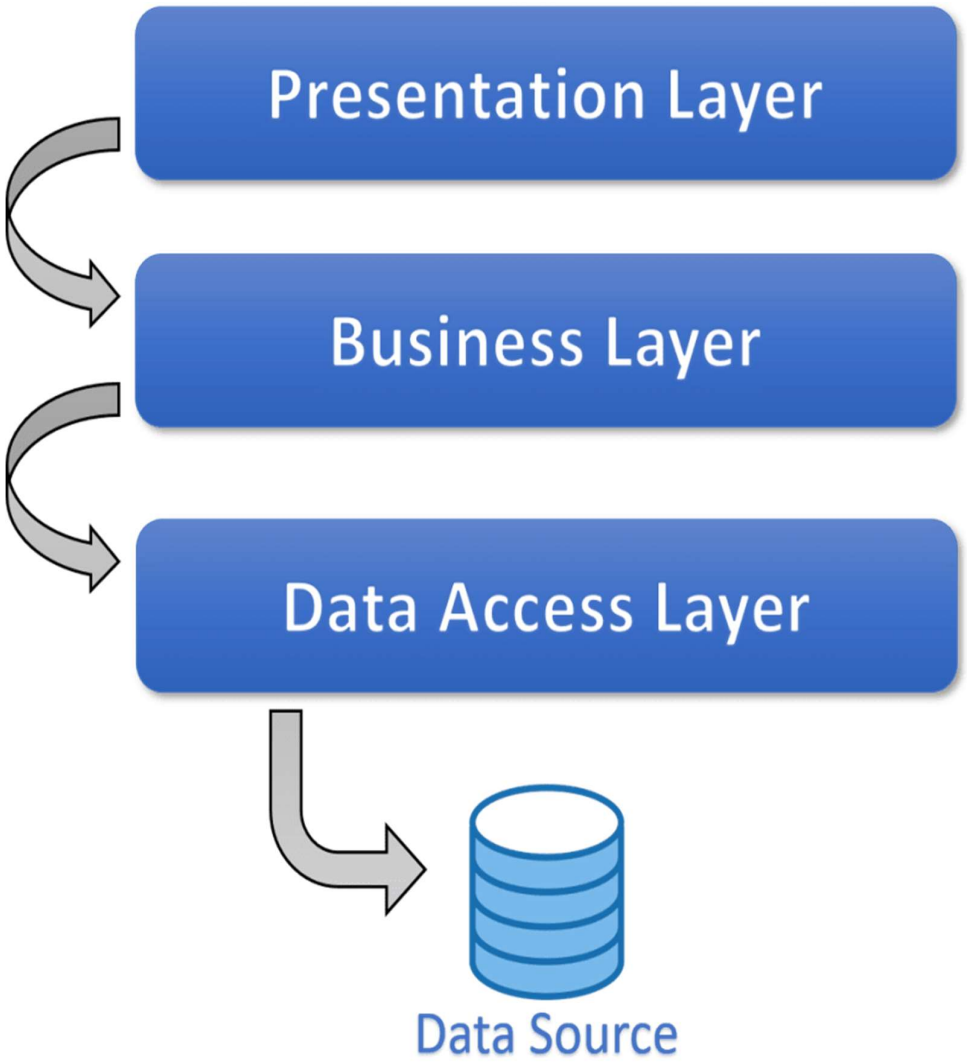

Figure 15: Three Layered Architecture

The three components of the architecture are as follows:

User Interface's Presentation Layer:

The user interface and user interactions are handled by the presentation layer.

It consists of elements including websites, mobile apps, desktop programmes, and other user interaction tools.

The user interface rendering, data collection, and output display are the

main goals of the layer.

In order to execute activities depending on user actions and retrieve data for presentation, it interfaces with the business logic layer.

Application Layer's Business Logic Layer:

The foundational features and regulations of the application are contained in the business logic layer.

It puts into practise the business procedures, algorithms, and guidelines that direct how the application behaves.

This layer manages interactions between the presentation and data storage layers by encapsulating and processing data, carrying out calculations, enforcing business rules, and orchestrating calculations.

The presentation layer sends requests to it, which it receives, processes, and then responds to with the relevant information.

Layer for Data Storage (Persistence Layer):

The management of data storage and retrieval is under the purview of the data storage layer.

It contains any other data sources the programme uses, such as databases, file systems, external APIs, etc.

Data persistence, querying, updating, and any necessary data transformations are handled by this layer.

By removing the underlying storage technology, it offers a means for the business logic layer to communicate with the data.

Dependency Injection

A software design pattern called Dependency Injection (DI) encourages loose coupling and allows for the separation of responsibilities across components of an application. Instead of generating or managing dependencies inside the objects themselves, it helps manage dependencies between objects by giving them the appropriate

dependencies from outside sources.

Micro Services

A collection of small, independent, loosely linked services is how an application is structured using the microservices architectural style. Each service represents a particular business capability and may be independently designed, implemented, and scaled. Modularity, flexibility, and scalability are encouraged by microservices, which makes it simpler to create and manage complex programmes.

Flexibility in Scaling

Each microservice's underlying app feature's demand can be scaled separately from the others.

This enables teams to correctly size infrastructure, ensure service availability during periods of heavy demand, and forecast feature costs with accuracy.

Simple Deployment

Microservices enable continuous integration and delivery, making it simple to test new ideas and roll them back if they don't work. More experimentation, quicker code updates, and a shorter time to market for new features are all made possible by the low cost of failure.

Utilisable Code

Software can be broken down into clear, separate modules that teams can use for a range of tasks. A service created for one function could be the inspiration for another feature.

2.1.5 Databases Migration

The process of moving information and structures from one database system to another is referred to as database migration. When an organisation needs to improve an existing database system, transfer to a different database platform, or combine many databases into one system, it is an important operation that they must complete. To guarantee data integrity and little downtime throughout the change, database migration requires rigorous planning, execution, and validation.

The justifications for moving a database can differ. Some businesses may opt to switch to a new database system in order to benefit from more sophisticated features, better performance, or more scalability. Others might be forced to migrate because of vendor changes, financial constraints, or legal restrictions.

2.2 Feasibility Study:

2.2.2 Technical Feasibility

Since Goland was used in the project's design, it is simple to install in any system as needed. It is more user-friendly, efficient, and simple for everyone to understand. PostgreSql can effectively handle enormous amounts of data and can handle stored procedures.

2.2.2 Time Schedule Feasibility

The project is straightforward to operate, and the fundamental requirements may be met in the allocated time period, satisfying the condition for time development feasibility. It was important to decide whether the deadlines were required or preferred.

2.2.3 Operational Feasibility

This application has a very large user base, and any user can use it to review their individual work. These kinds of applications for software engineer evaluation are spreading more widely every day. Additionally, this project is segregated into many modules, meaning that no user has access to view all modules; instead, they can only view the modules that they are authorised to view. Therefore, it is possible to operate this system.

CHAPTER 03:

SYSTEM DEVELOPMENT

3.1 Dataset Description

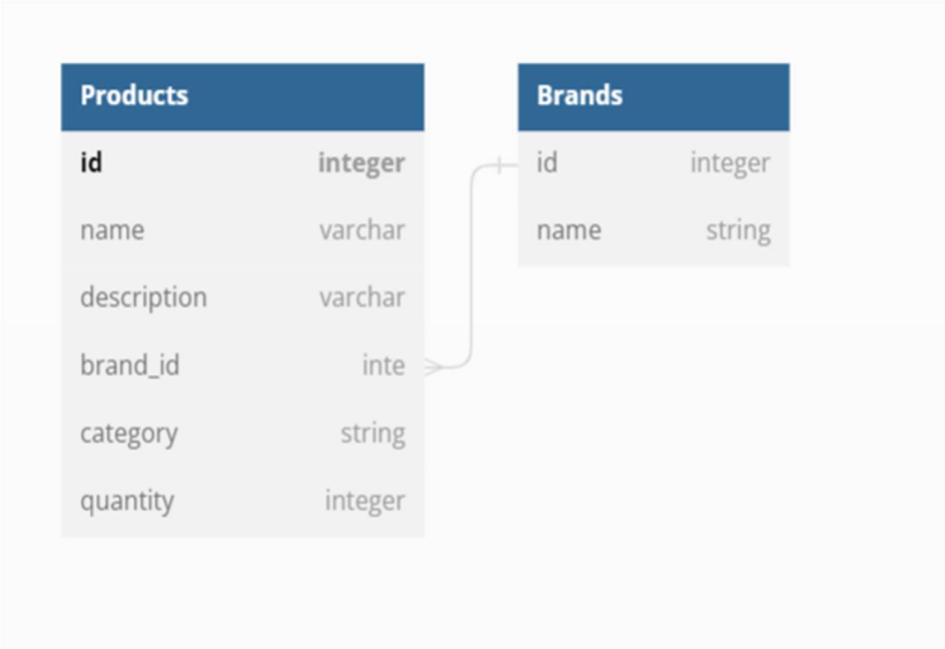

Figure 16: Database Schema

Database Tool: Mysql

Due to its simplicity of use, performance, and scalability, MySQL is an open-source relational database management system (RDBMS) that has grown significantly in popularity. MySQL AB, a Swedish business, developed it; Sun Microsystems later bought it; and Oracle Corporation now owns it. The preferred option for many web applications, content management systems, and data-driven websites, MySQL is widely utilised across a variety of industries.

The ease of use and simplicity of MySQL's interface are two of its main advantages. Both novice and expert users can use it because of its simple installation procedure and well-documented set of tools and commands. Since

the SQL-based language used by MySQL is well-known and comprehended, developers can write and run queries quickly.

The high performance of MySQL is well known. It is a dependable option for applications that handle a lot of read and write operations because it is optimised for processing massive datasets efficiently. Its speed and responsiveness are aided by the support for a number of indexing strategies, caching systems, and query optimisation capabilities. In order to boost scalability and high availability, MySQL also provides replication features, enabling businesses to make multiple copies of their databases.

Another noteworthy aspect of MySQL is its scalability. Without losing performance, it can manage expanding workloads and support growing data quantities. In order to enable horizontal scaling and distribute the workload across different servers, MySQL provides a variety of clustering and sharding techniques. Because of this scalability, businesses are able to manage more concurrent users and quickly growing data needs.

3.2 Project Development Approach and Justification

Approach to Project Development: Iterative Waterfall Model

This model separates the cycle into the following phases:

1. Possibility Analysis
2. Analysis and formulation of requirements
3. Unit testing, coding, and design
4. System testing and integration
5. Maintenance

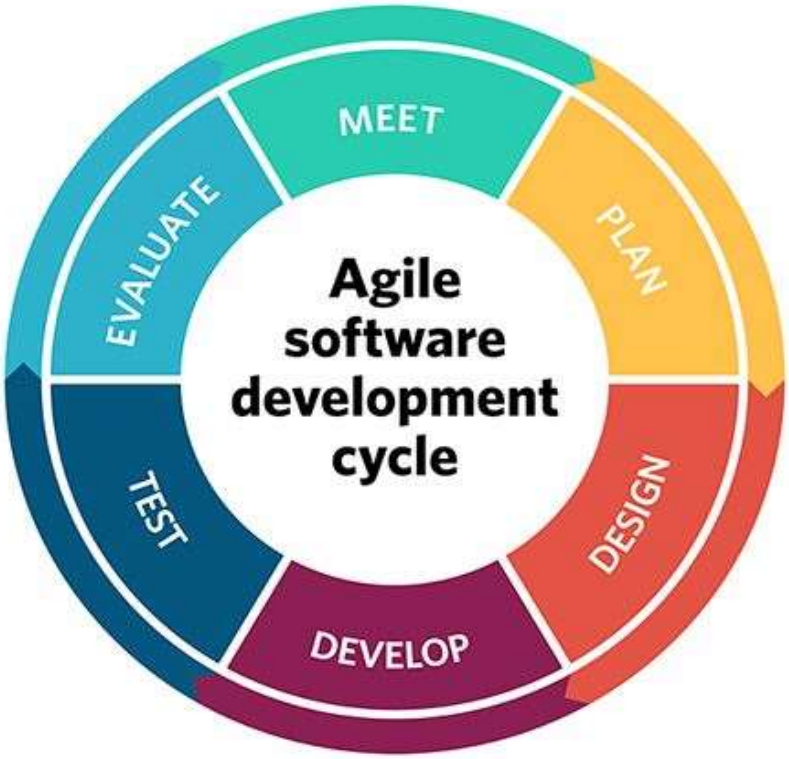

Figure 17: Development Approach (AGILE)

Agile is a project management and development methodology that places a strong emphasis on adaptability, teamwork, and iterative development. It came about as a reaction to the shortcomings of conventional waterfall approaches, which frequently led to rigid plans, sluggish feedback, and challenges adjusting changing requirements. Agile, on the other hand, encourages flexible planning, ongoing development, and strong cooperation amongst cross-functional teams.

Iterative and incremental development is one of the fundamental tenets of Agile. Agile divides a project into smaller, more manageable units called sprints rather than aiming to define and deliver the complete thing at once. The goal of each sprint, which typically lasts one to four weeks, is to produce a working product increment. This iterative technique enables teams to collect feedback, allows for frequent inspection and customization,

In Agile techniques, communication and collaboration are essential. Agile teams are frequently cross-functional and made up of members with a range of abilities. Daily stand-up meetings, when team members share progress, tackle problems, and coordinate efforts, build collaboration. Although direct communication is preferred, dispersed teams can communicate effectively by using virtual collaboration technologies. Agile fosters teamwork, facilitates knowledge sharing, and increases the team's collective intelligence by fostering a shared understanding of the project goals.

Customer engagement is a key component of Agile. It actively incorporates consumers or stakeholders throughout the development process while acknowledging that customer demands and priorities may change over time. Agile teams can gain important insights, verify hypotheses, and make necessary corrections to guarantee the final product meets customer expectations by participating in continuous feedback loops. High levels of client satisfaction are fostered by this customer-centric approach, which also makes it easier to supply solutions that closely match their needs.

Frameworks and procedures are provided by agile techniques like Scrum and Kanban to help with the application of these ideas. For instance, Scrum specifies the roles of the Product Owner, Scrum Master, and Development Team as well as the rituals of daily stand-ups, sprint planning, sprint review, and sprint retrospective.

Continuous improvement is also emphasised by agile approaches. Teams review their performance, pinpoint areas for development, and make changes to improve output and quality through routine retrospectives. Agile helps teams to effectively respond to changing requirements, developing technology, and altering market situations by promoting a culture of learning and adaptation.

Agile offers a flexible and team-based method for managing and developing projects. It is ideally suited for complicated and dynamic projects because of its iterative structure, emphasis on client value, and emphasis on continual improvement. Organisations can improve their capacity to provide high-quality

products, better meet customer expectations, and react to changing business environments by adopting Agile principles and practises.

3.3 Milestones and Deliverables

- Phase of feasibility analysis: one week
- Analysis of requirements and specification phase: 1 week
- Designing phase: 3 weeks approximately
- Phase of coding: about three weeks
- Each form is linked to a specific piece of database information.
- Phase of testing: around 15 days
- Analyses and tests were conducted on all required modules.
- With three tests, every feature operates and navigates properly.

3.3.1 Roles and Responsibilities

All phases were broken down into modules, with each module being given to a different member of the team because the project development was being handled by a single person. A job must be completed according to the specified parameters before the entire project is integrated.To complete the design of the entire application in the allotted time, I separated my work into numerous sections.

3.3.2 Group Dependencies

The following duties are dependent on one another:

Even though analysis or a system requirement study (SRS) is independent of all other work, it will begin once the feasibility study and project planning are finished.

Prototyping can be done concurrently with system analysis.

Prior to the project's development, a prototype's design and a system analysis

are conducted.

Only until certain key functionality have been developed and are ready for testing can testing begin.

3.3.3 Study of Current System

Examining and evaluating the state of RESTful API implementations across various businesses and domains is part of the research of contemporary REST APIs. Due to its simplicity, scalability, and interoperability with the web, REST (Representational State Transfer) has become a frequently used architectural approach for developing web services.

Researchers and analysts assess many facets of API design, implementation, documentation, and usage to investigate contemporary REST APIs. They evaluate how well REST principles—such as statelessness, resource identification, and standard interfaces—are adhered to by APIs. Additionally, they look at API endpoints, HTTP methods, request/response formats, and error handling procedures for consistency and clarity.

Researchers also look on the general developer experience and usability of REST APIs. The accessibility and quality of code examples, the clarity of API versioning and deprecation policies, and the ease of finding and comprehending API documentation are all evaluated. Reviewing the API's capabilities for rate restriction, caching, and authentication and authorisation protocols may be part of the investigation.

Analysing performance and scalability is essential to understanding REST APIs. Researchers test the performance of API calls in terms of response times, throughput, and latency under diverse user scenarios and loads. To ensure optimum speed and scalability, they evaluate the efficacy of caching systems, load balancing techniques, and API rate restriction rules.

A crucial component of REST APIs is security, which is also looked at in the study.

Researchers examine how security measures are implemented, including

authentication methods (like OAuth, JWT), secure data transmission (like HTTPS), and defence against common web vulnerabilities (like CSRF, XSS). They also look into whether the design and use of the API include appropriate access restrictions and data protection safeguards.

When analysing REST APIs, interoperability and integration potential are crucial factors to take into account. Researchers look at the availability of client libraries or SDKs for well-known programming languages, the support for data formats like JSON and XML, and the existence of standardised protocols and specifications like OpenAPI (formerly known as Swagger) for API description and discovery.

Researchers often perform surveys, interviews, or usability tests with API developers and users to acquire insights into the user experience and satisfaction. They elicit input on the difficulties, difficulties, and areas for improvement in using the API. Finding best practises, patterns, and guidelines for creating better REST APIs is made easier with the help of this user-centric approach.

The body of knowledge in API design and development is enriched by studying current REST APIs. In the API landscape, it assists in identifying trends, obstacles, and new practises that can direct API providers in enhancing their products. Additionally, it helps programmers choose and use solid REST APIs for their applications, thus raising the calibre, usability, and interoperability of web services.

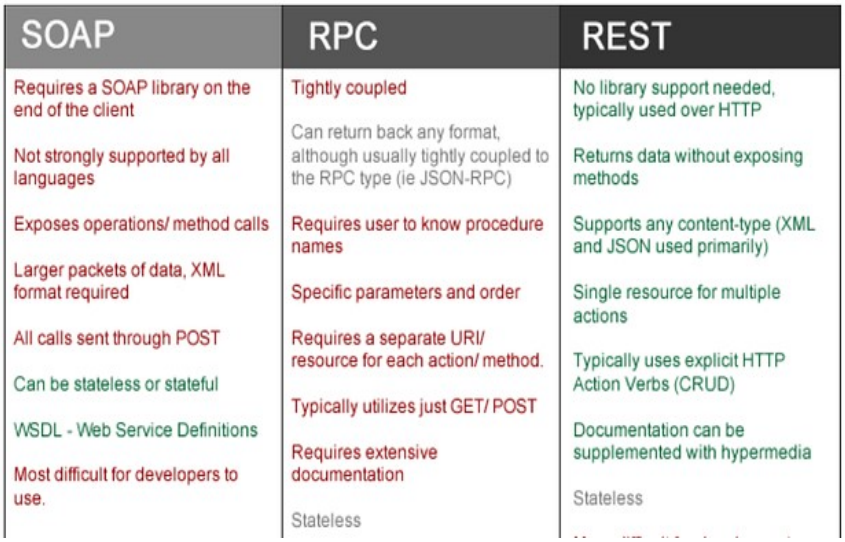

Figure 18: APIs

3.3.4 Problems and Weaknesses of Current System:

Although the architectural style known as REST (Representational State Transfer) has grown significantly in favour, it is not without its difficulties and restrictions. REST frequently causes issues, some of which are as follows:

- Lack of standardisation: Unlike SOAP (Simple Object Access Protocol), REST does not have a standardised specification. While this flexibility encourages innovation and personalization, it can also result in inconsistent API implementation and design across various services, making it more difficult for developers to comprehend and integrate with new APIs.

- Data over- and under-fetching are common problems with REST APIs since they often expose pre-defined endpoints that return set data structures. This might result in over- or under-fetching problems, where the API response either includes more data than is necessary or does not offer enough information. This causes inefficient network utilisation and extra processing on the client-side.

- Limited support for real-time communication: Since REST is primarily built on stateless, request-response interactions, it might not be appropriate in situations where real-time communication or bidirectional data flow are necessary. It is frequently necessary to use extra technologies or protocols, like WebSockets or long-polling, to provide real-time features like chat, live updates, or push notifications.

- Lack of discoverability: Without enough documentation and standardised methods for API description and discovery, discovering and utilising REST APIs can be difficult. While there is a solution in the form of tools like OpenAPI (previously Swagger), not all APIs follow these standards, making it more difficult for developers to locate and comprehend the accessible endpoints and their capabilities.

- Security considerations: Although REST provides secure connection via protocols like HTTPS and permits authentication and authorization procedures, the API developer is primarily responsible for putting security measures in place. If not effectively addressed, this decentralised method can result in inconsistent security implementations, raising the potential of vulnerabilities.

- Versioning and backward compatibility: As APIs change over time, it can be difficult for REST to retain backward compatibility while adding new capabilities or changing old ones. A smooth transition requires careful versioning methods and communication with API consumers because changes to the API structure or behaviour may disrupt current client applications.

- Lack of transactional support: REST does not have built-in support for atomic operations or distributed transactions over different resources since it is not intrinsically transactional. In complicated activities, ensuring data consistency and reliability frequently necessitates additional coordination or compensating methods that go beyond what REST is capable of.

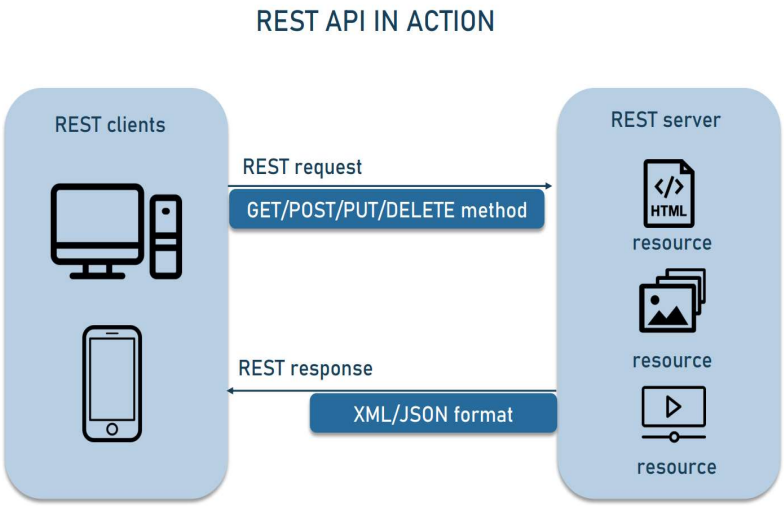

Figure 19: REST in action

3.3.5 User Characteristics

One system user primarily needs access to the system.

Administrator of Systems

- Permit adding Product and Brand Information
- Permit updating all product and brand information
- Permit seeing of all brands and products on the det

3.4 Hardware and Software Requirements:

3.4.1 Hardware Requirements:

• Processor: I3 (or)

• Higher Ram: 1 GB (or) Higher

• Hard disk: 10 GB (or) Higher

3.4.2 Software Requirements:

• Technology: Go Language

• IDE: GoLand

- Designing Tool: Creatly, Lucid Chart
- Server-Side Technologies: Go Lang
- Database Server: MySql
- Operating System: Linux, Mac

3.5 Assumptions and Dependencies:

The following assumptions are listed: -

Users are somewhat aware of how the system works, and the server is functional. Results from database changes are accurate and as expected.

Following is a description of dependencies: - The system requires a WiFi or Internet connection.

System Evaluation
Need for a New System:
Case Study Diagram

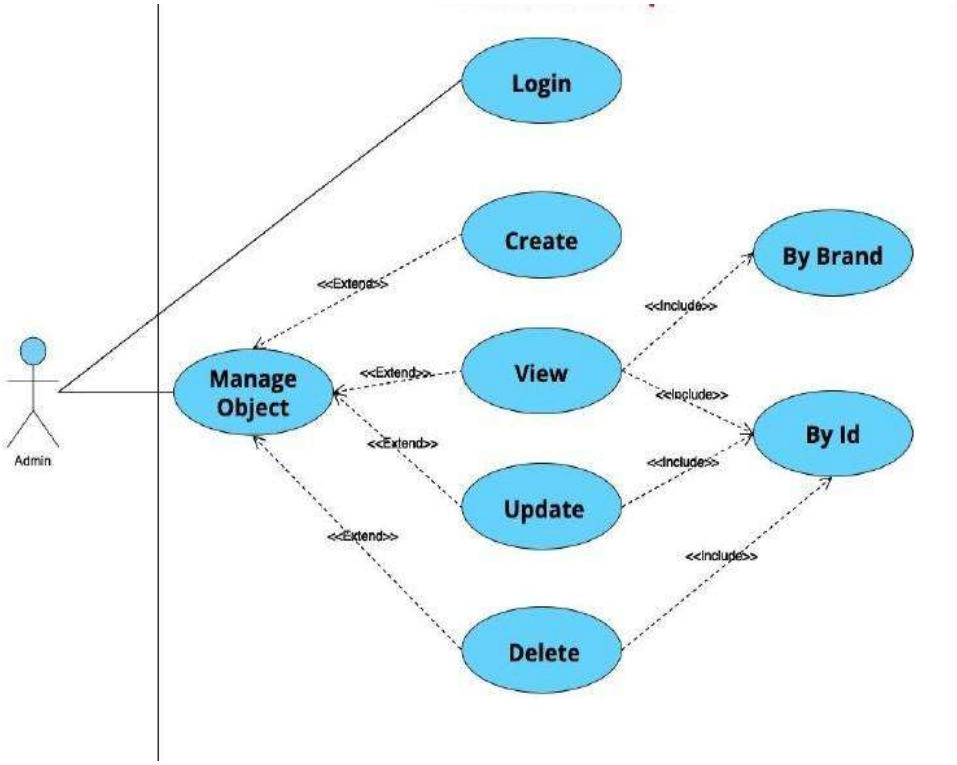

Figure 20: Use Case

This Figure is a use case diagram created to illustrate the system of the Product-Brand Store. It illustrates the workflow of any online store.
Basically, utilised for storage functionality, a use case diagram

R1: Create: Adding a new product and its reputable brand to the database.

Enter the following information about the product: name, description, price, quantity, category, brand name, and status.

Data entry into the database is processed.

output: database data inserted

R2: Delete: Erase an item based on its ID from the database

Enter: Id

Processing: Delete the database's data

data deleted from database output

16

R3: Update: Using the item's id, updating the database

Enter: Id

updating the database with new data

Data updated in database as output

View all the Products and Their Brands in

R4: Processing: Obtaining from the database all of the products and their brands.

Get the entire list of the products in the database as output.

R5: View Particular Product Details: View Particular Product with Brand

Input: Name /Id

Processing: Getting Product with their Brand according to input

Output: get product with their brand details

R6: Login: For Login into system

Input: Enter Username & Password.

Processing: Validate credential

Output: If all details is valid then login successfully

R7: Registration: for new registration

Input: Enter Username, Email id , Password .

Processing: Validate Details

Output: If all details is valid than register successfully

Get the database's whole list of products as an output.

Non-Functional Conditions:

- Security

When saving sensitive data to the database, it will first be encrypted. Only authorised administrators shall have access to the system's back end servers.
Responsive

The API must respond quickly. Therefore, the fully portable component should work with any system and any web browser and should be able to access all system functions, including those of any current or future hardware platforms. The system needs to function on PCs, laptops, etc.

- Maintainability

The developers can easily maintain the system because they have full access and access credentials.
Accessibility

The authorised developer has access to the system at any time and from any location.

- Usability

This approach is crucial for giving developers the satisfaction they desire in accordance with their needs and capabilities.
Mistake Handling

The system ought to prevent incorrect data entry. If the user provides incorrect inputs when entering the necessary data, the system should be able to alert various error messages to them.

3.6 System Design

3.6.1 Sequence Diagrams:

This flow chart begins when a user accesses a login page. The user clicks the login button after entering their username and password. The database receives a message from the login page containing the user login information to be verified. Two potential routes become apparent when the username and password have been verified by the database (alternative fragment). The database notifies the user that their login has been accepted if the login information is accurate. The system notifies the user if the login information was wrong. The password field is cleared by the system. The user can then reenter the password.

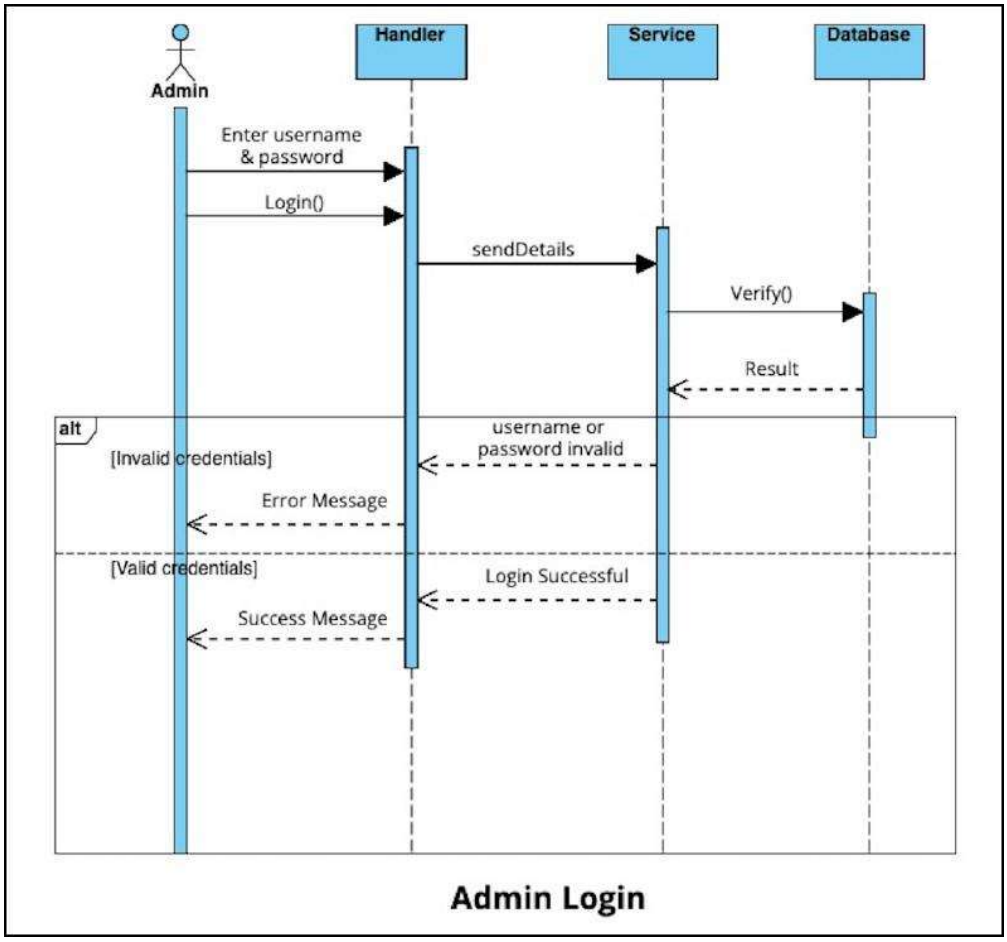

Figure 21: Sequence Diagram of Login

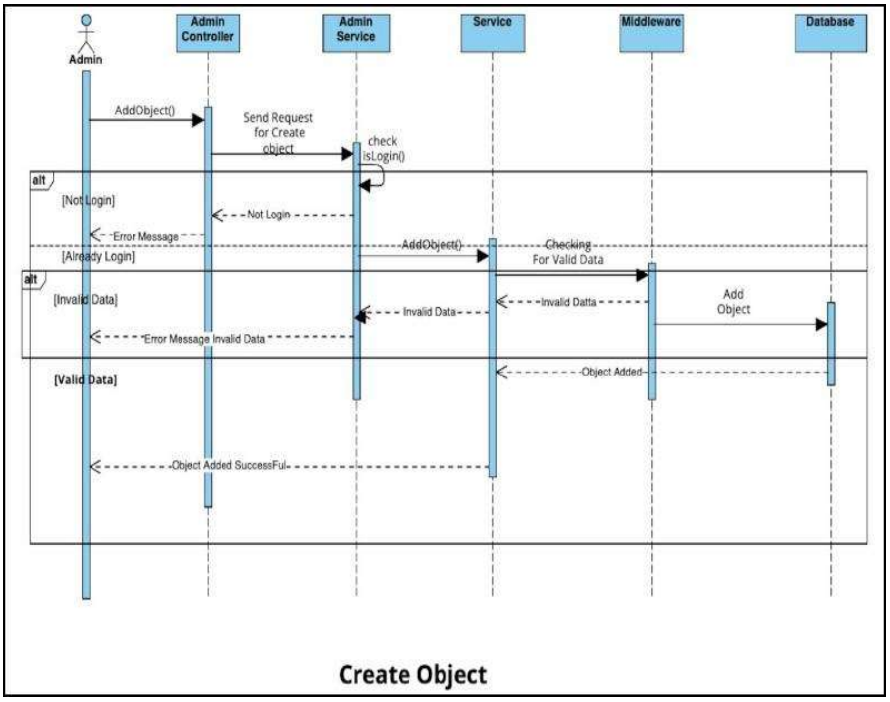

Figure 22: Sequence Diagram of Create Object

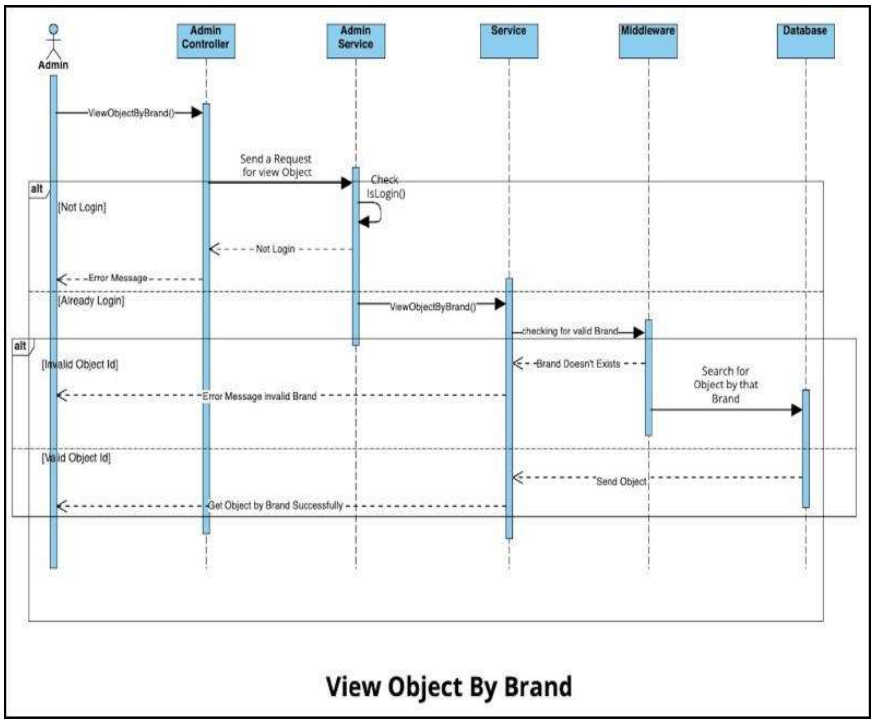

Figure 23: Sequence Diagram of GetByBrand

3.7 Implementation Environment:

For implementation, we used the programming language Golang.
as a Database MySql

Specifications for Modules

the admin module

Coding Guidelines

We were quite careful when writing our code to adhere to the fundamental coding best practises for Golang, such as the limited use of global variables.

observing proper naming practises for constants, functions, global variables, and local variables.
accurately indenting.

proper handling of errors.

Making comments to aid comprehension.

CHAPTER 04:
EXPERIMENTS AND RESULT ANALYSIS

4.1 Test Plan

Black box testing will be the testing approach employed in the project. Black box testing involves applying the application's anticipated inputs but only checking the results of those inputs.

In the Go programming language, testing is a crucial step in the creation process and is strongly encouraged. Developers may easily build tests in Go to check the quality and correctness of their code thanks to the language's powerful and integrated testing framework. Go's testing package provides an easy-to-use syntax for building tests that enables you to declare test functions, make test cases, and make assertions. The Go testing framework encourages the practise of creating short, targeted tests that target particular units or capabilities in order to quickly find and isolate problems. Developers may easily run tests and provide illuminating test findings thanks to the go test command and the testing package. putting a big focus on testing

4.2 Testing Strategy

- **UnitTesting**.

  In order to assure the accuracy and dependability of individual units or components of code, unit testing is an essential practise in software development. It focuses on controlled and repeatable isolation and evaluation of small units of code, such as functions, methods, or classes. Developers can find and fix bugs early in the development process by testing units separately from the rest of the system. A continuous integration or build procedure often and automatically runs unit tests as part of it.

- **Linters check**

  Static code analysis, commonly referred to as linter checks, is a crucial technique in software development for spotting possible problems, enforcing coding standards, and fostering code quality. Linters scrutinise source code to find recurring programming errors, potential bugs, style infractions, and other troubling trends. Linters can offer developers insightful comments and suggestions to enhance the codebase by applying a set of established rules or unique configurations.

  A project or organisation can benefit from linter checks to maintain standard coding practises. They uphold coding rules including indentation, name conventions, and formatting styles to make sure the source has a consistent and readable structure. This consistency improves teamwork and makes the process of code review easier.

  Linters also have the ability to identify potential flaws and programming mistakes early in the development process. They can spot problems like unused variables, variables that haven't been initialised, inappropriate error handling, or incorrect function calls. Linters assist in lowering the possibility of running into runtime issues, enhancing code reliability, and cutting down on debugging effort by drawing attention to these issues during development.

4.3 Unit Testing

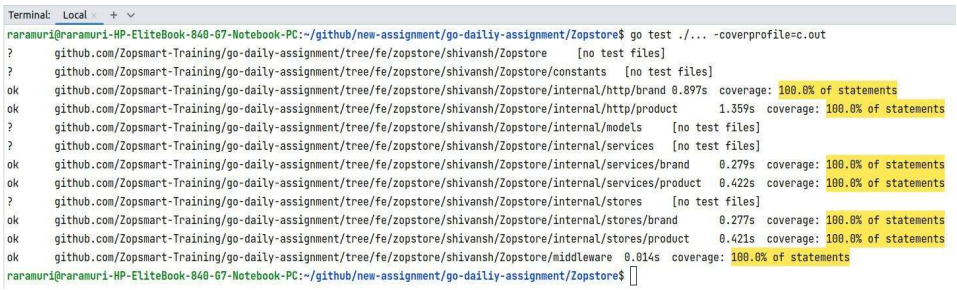

Figure 24: Test Result Coverage

checking to see if there are any Code Walk-thru reports available that have proven the presence of and adherence to coding standards.

Examining unit test specifications

Check that the programme specifications and the unit test specifications are compatible. • Check to make sure all boundary and null data conditions are present.

Covering all programme paths during testing (using white-box testing) is the best technique to ensure that every control flow is taken care of. This indicates that for a "if" statement, both branches are exercised, for a case statement, all branches are exercised, for a while statement, the loop is taken once, several times, or not at all, and for complex logical expressions, all components are executed. Path Testing is what we call this. The Boolean expression's entirety is reported by Branch Testing.

assessed to both true and false when tested in control structures.

It also covers the use of switch statements, exception handling, and interrupt handling. Since it takes into account every potential combination of individual branch conditions, path testing also includes branch testing. Statement testing, a less complex variation, checks to see if each programme statement has been run at least once.

4.4 Test Cases

Table 1: Login Test Cases

Test Object Descriptions	Test Conditions (Input)	Expected Result	Status Code	Actual Result	Pass/ Fail
Login	Email is not correct	Validation error	404	As Expected	Pass
	Password is not correct	Validation error	404	As Expected	Pass

Table 2: Test Cases for GetbyId

Test Object Descriptions	Test Conditions (Input)	Expected Result	Status Code	Actual Result	Pass/ Fail
Get By Id	Valid Id	View Object for particular Id	200	As Expected	Pass
	Invalid Id	Error Message Invalid Id	404	As Expected	Pass
	Empty Id	Error Message Please Enter Id	400	As Expected	Pass

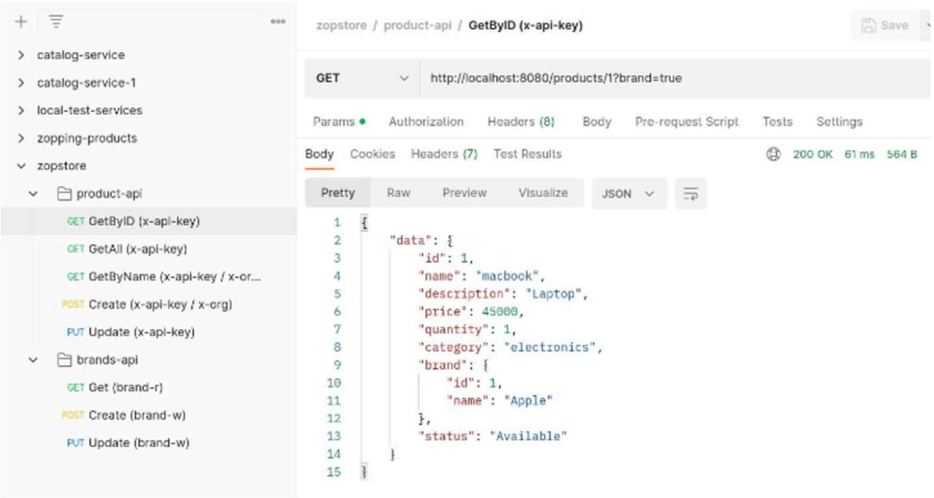

Figure 25: Postman GetById

Table 3: Test Case for Create Object

Test Object Descriptions	Test Conditions (Input)	Expected Result	Status Code	Actual Result	Pass/ Fail
Delete	Valid Id	Object Created Successfully	201	As Expected	Pass
	Invalid Brand	Error Message Invalid Brand	404	As Expected	Pass
	Empty Id	Error Message Please Enter Id	400	As Expected	Pass

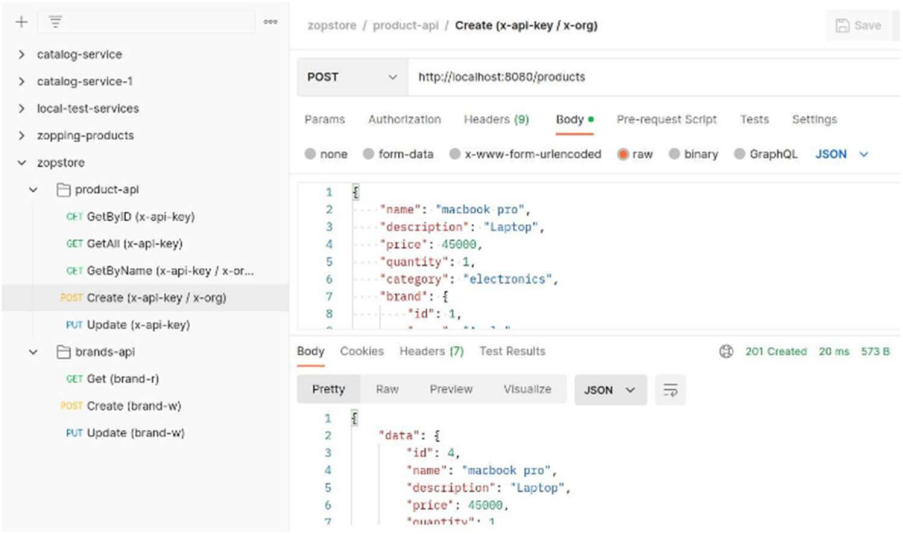

Figure 26: Postman for Creation

Table 4: Test Case for Update

Test Object Descriptions	Test Conditions (Input)	Expected Result	Status Code	Actual Result	Pass/ Fail
Update Object	Name **or** Brand **or** Description **or** Price **or** Quantity **or** Statuts	Object Updated Successfully	200	As Expected	Pass
	If Id == 1002	Invalid Id	400	As Expected	Pass

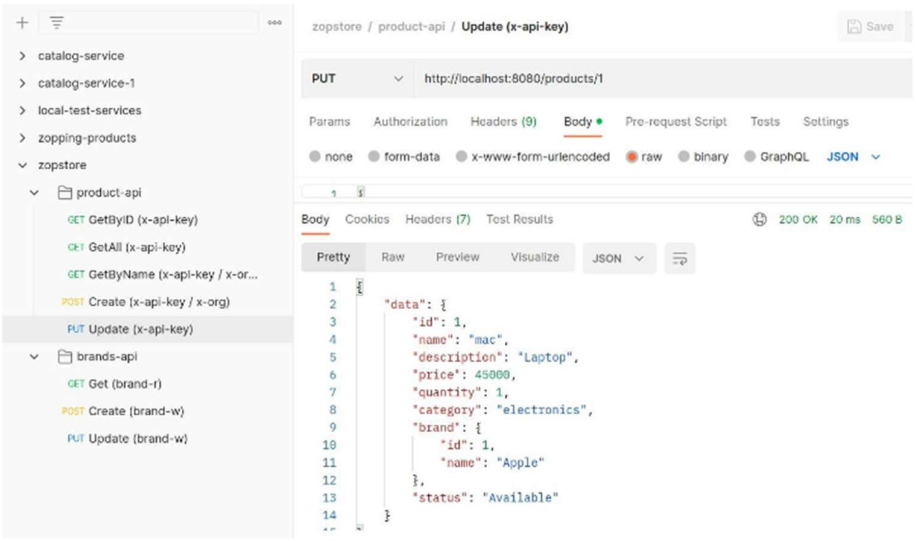

Figure 27: Postman for Update

CHAPTER 05:

CONCLUSION

5.1 Conclusion

It was astonishing how much I learned while working on this project. Working on this project allowed me to go through every stage of the project's development, which gave me a newfound appreciation for the field of software engineering. Through the joy of working and the thrill of conquering various challenges, I developed an understanding of the development industry. As a result of this project, I discovered how expert software is produced.

In this project, we developed a dependable, easy-to-use, cost-effective, and useful system to handle product and brand information.Owners may thus control the specifics of their internet business with ease and effectiveness. It saves the owner a tonne of time and money.

In conclusion, REST API training is a worthwhile investment for people and businesses wishing to improve their knowledge and skills in creating and utilising web services. Participants receive a profound understanding of REST principles, architecture, and best practises through thorough training. They acquire the skills necessary to create, create, and test RESTful APIs, guaranteeing compliance with industry standards and fostering interoperability.

Participants in REST API training leave with the skills and tools needed to build reliable and scalable APIs. Important ideas like resource identification, HTTP methods, request/response formats, and error handling are included in their education. Additionally, training covers subjects like versioning, security, and authentication to help participants develop safe APIs.

Participants gain hands-on experience creating RESTful services by participating in practical activities and real-world scenarios. They get knowledge about how to design, document, and test APIs using frameworks and tools. With the help of this practical knowledge, they can speed up the creation of APIs, increase productivity, and provide users with high-quality APIs.

Furthermore, by highlighting the significance of creating APIs with the end-user in mind, REST API training supports a customer-centric strategy. Participants learn about the best practises for API documentation, discoverability, and usability. They get knowledge on how to implement feedback loops and involve stakeholders at every stage of the lifetime of an API to guarantee that the APIs they design are user-friendly, thoroughly documented, and offer a superior developer experience.

In general, REST API training equips students with the expertise needed to succeed in the quickly developing industry of web services. It enables them to create interoperable, scalable, and secure APIs that support seamless integration, innovation, and digital transformation. Whether people want to succeed in their jobs or businesses want to improve their API programming capabilities.

5.2 Discussion

We began the project by becoming familiar with the fundamentals of Golang, Git/Github, Postman, and MySQL. Next, we were familiar with the workflow of the software development life cycle before moving on to implementation. I spent time learning each of these things. This WEB API's implementation took a significant amount of time. I next tested the entire Web API. I felt confident that once that was finished, the project would be completed successfully.

5.3 Future Scope and Enhancements

REST API's potential future uses are expanding as technology progresses. The following are some crucial sectors where REST API is anticipated to expand and have an impact:

Microservices Architecture: The REST API is essential to the creation and fusion of microservices. The focus of the microservices design is on dividing large, complicated programmes into smaller, autonomous services that can communicate with one another via APIs. Microservices can communicate and share data using RESTful APIs, which allows for the construction of flexible and modular applications.

Internet of Things (IoT): As the IoT ecosystem grows, REST API will be essential for facilitating data exchange and communication between linked systems and devices. RESTful APIs offer an industry-standard and standardised way for IoT devices to communicate with software and services, enabling seamless integration and the development of creative IoT solutions.

REST API is essential for creating mobile applications that communicate with backend systems, according to the field of mobile application development. RESTful APIs act as a link between mobile apps and the server-side architecture as mobile devices become more and more common. Through well defined APIs, they enable mobile apps to make transactions, obtain data, and use a variety of services, assuring effective communication and improving user experiences.

Machine learning (ML) and artificial intelligence (AI) applications need to integrate REST APIs in order to access and use data or services from outside sources. RESTful APIs are frequently used by AI and ML models to receive data, do computations, and report results. Integration of REST APIs facilitates fluid communication between AI/ML systems and other applications, encouraging the creation of smart, data-driven solutions.

Integration with Third-Party Services: RESTful APIs provide a framework for integrating and communicating with third-party services, including payment gateways, social media networks, mapping applications, and more. Integration of REST APIs with external providers offers smooth communication, data sharing, and functionality as organisations continue to use other services to improve their products.

RESTful APIs are necessary for integrating decentralised applications with blockchain networks while working with blockchain technology and decentralised applications (DApps). They make it possible for developers to carry out transactions, access blockchain data, and communicate with smart contracts. Integration of REST APIs makes it easier to create safe,

decentralised applications that take advantage of blockchain technology's advantages.

Security and Authentication: REST API will continue to progress in terms of providing sophisticated security methods as the demand for secure and authenticated communication rises. To achieve strict security standards, this entails integrating strong authentication mechanisms, like OAuth, and guaranteeing data privacy and encryption.

In conclusion, the REST API has a bright future with a variety of uses and rising opportunities. It is well-suited for a variety of emerging technologies and trends because of its flexibility, scalability, and interoperability, including microservices, IoT, mobile applications, AI/ML, third-party service integration, blockchain, and security. REST API will continue to develop and serve a crucial role in enabling seamless communication, integration, and innovation across various areas as technology develops.

5.4 Application Contributions

Below are a few open source and real-world applications to which GoLang has contributed.

- The main Python components of Docker Dropbox, a container management system and set of tools for deploying Linux containers, were converted to Go.

- Using the go-ethereum implementation of the Ethereum Virtual Machine, Ethereum is a blockchain for the digital currency Ether.

- A web-based platform called Gitlab for the DevOps lifecycle includes a Git repository, a wiki, tools for tracking issues, continuous integration, deployment pipelines, etc.

5.5 Limitations

1. Lack of Flexibility: Simple APIs frequently lack functionality and may not provide the necessary flexibility to accommodate a variety of use cases. They might only offer a predetermined set of functions or endpoints, making it difficult to adapt or add functionality to meet particular needs.

2. Limited Scalability: Simple APIs might not be able to manage high numbers of concurrent users or requests. They could be deficient in features like rate limitation, caching, or load balancing, which are crucial for scaling an API to manage heavy traffic and guarantee peak performance.

3. Simple APIs may not have adequate security safeguards, making them susceptible to intrusions, data breaches, and other security risks. The integrity and confidentiality of data may be jeopardised if features like authentication, authorization, and encryption are absent or improperly implemented.

4. Insufficient Error Handling: Simple APIs may only offer a limited number of error handling options, making it difficult to troubleshoot and diagnose problems. They might not deliver in-depth error messages or insightful feedback that aids developers in figuring out the root of errors or offers advice on how to fix issues.

5. Limited Documentation: Developers may have trouble understanding how to use simple APIs because they lack thorough and current documentation. Insufficient documentation can delay integration, lengthen the development cycle, and prevent third-party developers from using the API.

6. Lack of Versioning: It's possible that simple APIs don't allow versioning, which is necessary for managing API upgrades and modifications without interfering with already-established client

integrations. Without versioning support, any changes to the API run the risk of interfering with already-running apps and creating compatibility problems.

7. Poor Extensibility: Simple APIs might not offer obvious ways to add new feature or extend existing capabilities. This constraint may make it difficult to add new features or modify the API to meet evolving needs, which could limit scalability and constrain the capacity to make future improvements.

8. Limited Monitoring and Analytics: Simple APIs might not provide thorough monitoring and analytics features. For locating problems, maximising performance, and making data-driven decisions for API upgrades, real-time monitoring of API usage, performance metrics, and analytics on user behaviour and trends are essential.

REFERENCES

1. https://www.restapitutorial.com/
2. https://swagger.io/docs/
3. https://cloud.google.com/blog/products/api-management/restful-api-design-best-practices-for-apis-part-1
4. https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design
5. https://www.oracle.com/technical-resources/articles/it-infrastructure/restful-api-design-best-practices.html
6. https://dzone.com/articles/rest-api-design-the-complete-guide
7. https://www.programmableweb.com/api-university/restful-apis/how-to-design-great-restful-apis
8. https://www.programmableweb.com/api-university/restful-apis/how-to-make-your-api-restful
9. https://golang.org/doc/
10. https://golang.org/doc/effective_go.html
11. https://gobyexample.com/
12. https://pkg.go.dev/std
13. https://www.tutorialspoint.com/go/index.htm
14. https://golang.org/ref/spec
15. https://github.com/golang/go/wiki
16. https://github.com/avelino/awesome-go
17. https://dev.mysql.com/doc/
18. https://www.tutorialspoint.com/mysql/index.htm
19. https://dev.mysql.com/developer/
20. https://dev.mysql.com/doc/refman/
21. https://dev.mysql.com/doc/workbench/
22. https://www.percona.com/blog/
23. https://www.mysql.com/high-availability/
24. https://stackoverflow.com/questions/tagged/mysql
25. https://docs.docker.com/
26. https://hub.docker.com/

27. https://www.docker.com/blog/
28. https://github.com/docker/docker
29. https://forums.docker.com/
30. https://www.youtube.com/user/dockerrun
31. https://stackoverflow.com/questions/tagged/docker
32. https://www.docker.com/customers/success-stories
33. https://kubernetes.io/docs/
34. https://kubernetes.io/docs/concepts/
35. https://github.com/kubernetes/kubernetes
36. https://kubernetes.io/blog/
37. https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/
38. https://kubernetes.slack.com/
39. https://stackoverflow.com/questions/tagged/kubernetes
40. https://kubernetes.io/docs/tutorials/