Industrial Internship Report

Training in "Software Development with Java"

Done at

ZopSmart, 2078, 24th Main Rd, Vanganahalli, 1st Sector,

HSR Layout, Bengaluru, Karnataka 560102.

by

Vaishnavi Singh (171243)

Under the supervision of

Mr. Ujjawal Misra

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan-173234

Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my work carried out under the supervision of **Mr. Ujjawal Misra** during the internship at **ZopSmart**.

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

Vaishnavi

Vaishnavi Singh (171243)

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

[illegible]

(Dr. P.K. Gupta)

Dr. Pradeep Kumar Gupta

Associate Professor

Computer Science and Engineering

Acknowledgment

With great pleasure I avail this unique opportunity to express my profound sense of gratitude and indebtedness to **Mr. Ujjawal Misra**, **Consulting Architect at ZopSmart**, for his effective advice, perpetual and a never-ending supply of motivation and constructive criticism during the course of the project. I would also like to express my sincere gratitude to **Mr. Nikhil R.S.S.** and **Mr. Shubham Joshi**, **Senior Software Development Engineers at ZopSmart** for guiding us throughout the course of internship, and other developers of this project group for their help rendered in completion of this project work. Additionally, I would like to thank **Dr. Pradeep Kumar Gupta(Associate Professor, JUIT)** for consistently guiding me during college on various projects.

I am also obliged to JUIT College and the staff members for the useful knowledge they provide in their respective fields during the time of my assignment. I am thankful for their cooperation.

Finally, I thank the Almighty for relentless support, our parents and our classmates, without whom this task would not have been possible.

Table of Contents

CHAPTER 3 : CONCLUSIONS

Abstract

ZopSmart is a retail technology company that offers all the resources needed to start and develop your online store. It has a range of items that can rapidly and efficiently inspire you to accomplish your objectives whether you are a traditional retailer looking to develop an omni-channel market or perhaps an online only retailer planning to develop an e-commerce business. It's solution is now in development since last seven years and processes thousands of transactions nearly everyday. Throughout the field of retail technology, our suite of products is among the most innovative, robust, and scalable solutions available. It allows a vendor to create an e-commerce website to handle their everyday business operations. Setting up own website using our platform is indeed a helpful way to learn more about it. Users can begin by including some items and banners to your site. A variety of themes are also available to choose from. There is one programme that creates static websites. This offers customers with a library of pre-built react modules with which one can pick and customise. An entire website can be created using a drag-and-drop gui. You can sign up once more if you haven't already. After a few minutes one should be able to put that to the test. Many of the modifications take effect right away. All of our knowledge has just been incorporated into market practises that become part of our brands. To give consumers an excellent shopping experience with insightful product search. For convenience of buying, it has created a personalised product catalog. Self-service rescheduling and returns are available. The Enterprise portal is used by two of India's biggest e-grocers to operate their online operations. The company makes use of Java, Javascript, Golang, Android and numerous other technologies to build products for retail. I did my training in the Java development field, this report covers the work done during my training in ZopSmart.

CHAPTER 1 : INTRODUCTION

1.1 Introduction to Ecommerce

Ecommerce, or electronic commerce or internet commerce, is the purchasing and selling of products and services over the internet, including the monetary and information transfers used to complete these transactions. Ecommerce is frequently used to refer to the online selling of physical goods, but it may also extend to any form of commercial transaction that is made possible by the internet.

Though e-business encompasses all forms of running an online business, ecommerce focuses on the exchange of products and services. A platform on the Internet is similar to electronic commerce. Electronic commerce is the process of shipping, ordering, selling, promoting, and serving goods and services using electronic equipment like the Internet and other computer networks. It E-commerIte is based on the same general concepts as conventional commerce: producers and consumers trade commodities and ship them between one location to the next.

Figure 1: How ecommerce feels like

1.1.1 Ecommerce Models Types

Four basic ecommerce structures which could be used to represent every other exchange amongst businesses and customers.

- B2C (Business to Consumer): When a company offers a product or service to a single customer. (e.g. Amazon)
- B2B (Business to Business): When a company sells a product or service to another company. (e.g, Shopify)
- Consumer to Consumer (C2C): Whenever a person sells a product or service to some other. (e.g, ebay)

- Customer to Business (C2B): When an individual sells their own goods or services to a company or organisation. (For example, an influencer can charge a premium for advertising to their online community, or a photographer may licence their image for commercial use.)

1.1.2 Ecommerce Case Studies

Different transactional arrangements between companies and customers, as well as different items being traded as part of such transactions, can all be found in ecommerce.

- Retail: The distribution of a commodity directly to an user by a company without any need for an intermediary.
- Wholesale: The selling of goods in large quantities to a manufacturer who then offers them to customers directly.
- Dropshipping: The selling of a commodity that is supplied by a third party and sent to the customer.
- Crowdfunding: The compendium of money from customers prior to the release of a good or service in order to boost the seed funding required to bring it to retail sector.
- Subscription: A fully automated repetitive purchase of an item or brand on a continuous basis till the registered user cancels.
- Physical products: A certain tangible good which necessitates replenishing stock and physically shipping product to customers as sales are made.
- Digital products: online products for download, templates, and training or media that have to be bought or licenced for consumption.
- Services: A expertise,skill or group of skills that is given for a charge. The period of the service provider may be bought for a fee.

1.1.3 The Components of Electronic Commerce

1) Goods or services: In the case of E-Commerce, a simulated product is displayed on the internet On some web page, someone could display a digital depiction of an item and its extensive set of features, which is not always available for actual goods used in business.

2) Location to sell goods: mostly in case of E-Commerce, this is a website that exhibits merchandise in a variety of forms and serves as an E-Commerce platform.

3) Method of attracting consumers to the internet site: Inside E-Commerce, search engines and links to other websites are an essential part of helping individuals in reaching E-organization portals.

4) Method of accepting yields: Order can be placed directly from the site. Shopping carts also available on the websites of such businesses. One should click the button and print out the shopping card to make a purchase for goods to be ordered, and the E-Commerce firm can approve that as a client request or order.

5) Receiving funds in this manner: Purchasers are in close touch with one another in conventional trade. Electronic money transfers, such as credit and debit cards, smart cards, and e-checks, are used to accept transactions in E-Commerce. Payment information is processed by Value Added Networks (VANS) and Payment Gateway Systems, among other technologies.

6) Method for dealing with warranty coverage: If the product is faulty or has any issues, it is often necessary to file a warranty petition. Warranty claims must be honoured in this scenario, just as they will in a commercial environment.

7) Method of customer support: E-mail, online forms, online knowledge bases, and commonly inquired questions are the core instruments of customer support.

1.1.4 E-commerce performance drivers

In certain instances, online retailing can thrive not just because of the product, but also because of its capable management staff, excellent after-sales facilities, well organised corporate strategy, internet infrastructure, and a stable, excellently designed website. A business that wishes to thrive would focus on two areas: technical and operational aspects, as well as customer service. Market study and review have been completed to a satisfactory level. E-commerce is not immune to sound market strategy and production and consumption principles. Company loss is as common in e-commerce as it is with any other industry.

1.1.5 The Benefits of E-Commerce

Indeed there are hundreds of explanations for why the country or literally anyone with network access, appears to be centering for e-commerce. Internet has benefits like increased coverage and second lower cost of operating. Both of these advantages will have a huge influence on the global market.

Listed are some major merits of e-commerce:

1. Procure More Clients

Stores on the internet get more exposure to a larger audience. Where a store online might be reached and used by an unlimited internet users, a physical one can only be accessible to local people. Several easiest channels to accomplish new consumers using strength of internet usually involve:

- Results from a Search Engine - If a platform is well optimised, business can attract a huge number of new consumers through search search engines Such as Google. More developed your content is, the better it will appear in results pages, and then the more potential users it will be able to draw.
- Sharing on Social Media - If your clients are pleased with your goods, social media makes it very easy for any of them to tell their relatives and friends about it. This works similarly to utterance in retail markets, and with the web, customers can get a clear relation to the item upon on website.

2. Quite widely available

The website, and hence e-commerce shop, remains open 24/7 a day, seven days a week, apart from a regular shop, no limitations of geography. Customers will have much more exposure to the shop as well as its offerings as a result of this. E-commerce could also be reached from any location with internet connectivity. Customers no longer have to drive to a place, which saves them energy / cost.

Figure 2: A 24/7 open store

3. Customer Insights

Data analytics for customers will help build anything from target markets to sell certain items. A regular store will let you be able to assemble several user records, but it'll cost time and even perhaps turns out ot be unreliable.

1.2 The challenge

There can be variety of challenges encountered in an ecommerce platform, the following are by far the most popular ecom platform development issues:

- Choosing the best development agency
- Selecting the appropriate technologies for your platform, such as a shopping cart and inventory control tools.
- Maintaining a technologically-advanced mindset
- Including insights and improve the user's experience (UX)
- Creating material that is both insightful and exclusive
- Creating appealing call-to-actions that maximise consumer conversion ● Reducing the time required for your platform to load.
- If a platform is not phone - friendly, visitors can abandon it before ever looking at the stuff.
- Management of increasing customers and their data.

CHAPTER 2 : SYSTEM DEVELOPMENT

2.1 Proposed Methods and Tools

2.1.1 Microservices

Microservices are an emerging software delivery model in which programme code is served in limited, smaller portions that are separate from each other. Because of their limited size and higher separation, they may have a range of additional advantages, including smoother servicing, increased efficiency, increased fault tolerance, better market coordination, etc. The ‘new standard' is microservice models. Making lightweight, self-contained, fully prepared apps will give the program more versatility and durability. Multiple purpose-built functionality in Spring Boot make this simple to create and operate microservices in production at scale.

2.1.1.1 Microservices and monolithic architecture

A monolithic programme is made up of a single piece of software. A repository (comprising several tables in some kind of a dbms), a consumer ui (comprising of Webpages and/or JavaScript working in a google chrome), and a server-side framework make up an enterprise solution. This server-side programme can perform HTTP requests, execute domain-specific logic, download and edit information from the database, and generate HTML frames for browser delivery. It's a single logic program meaning a monolith. A programmer should develop and deploy a modified version of the server-side programme to allow further changes to the framework.

Microservice features, on the other hand, are formally articulated by business-oriented APIs. They remain important commodities to the company because they encompass a key market skill. Since the application is specified solely in commercial terms, the development of service, that could include integrations with databases, is totally concealed. The placement of resources as important resources to the company implies that they are suitable for a variety of situations.

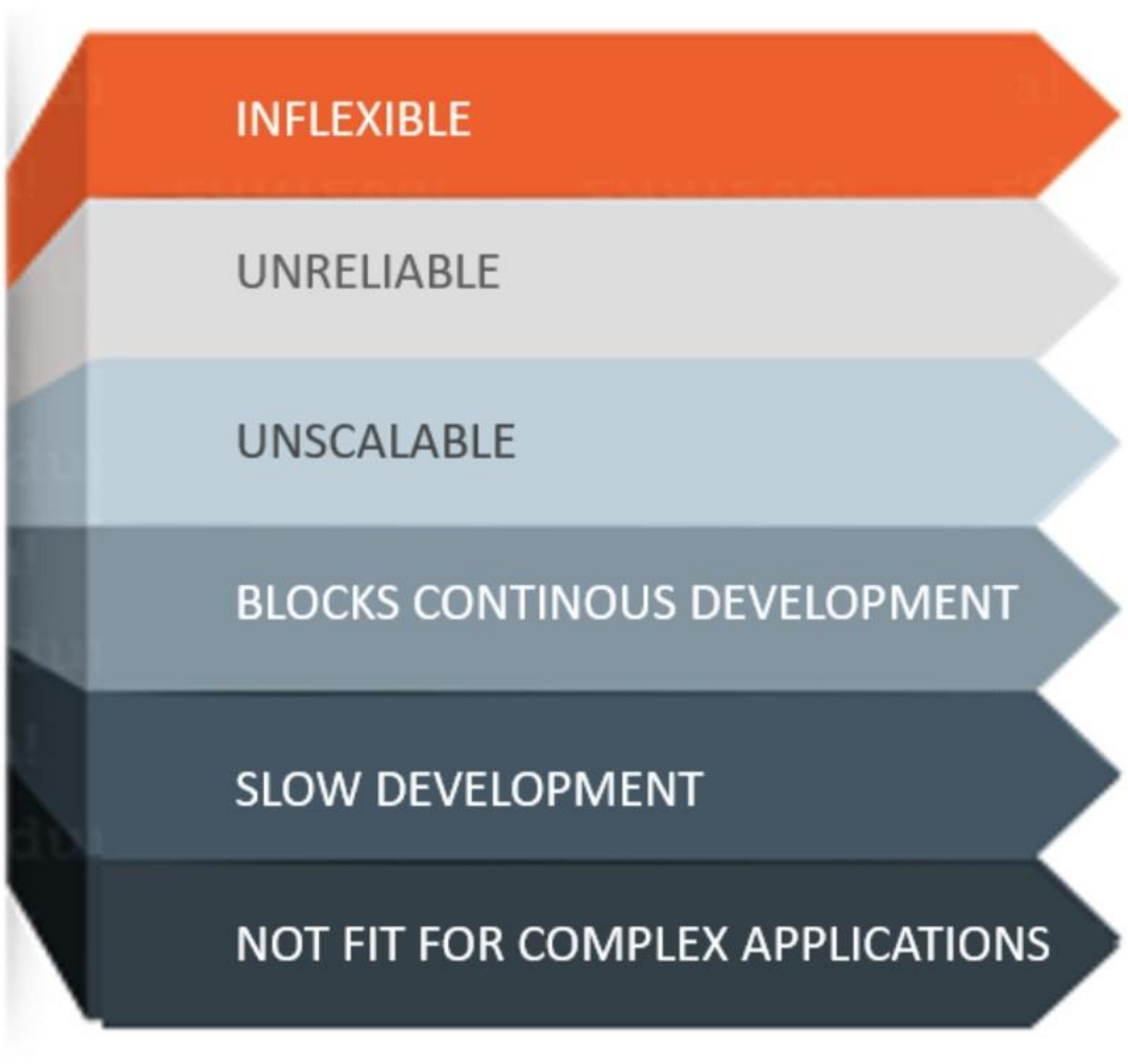

INFLEXIBLE
UNRELIABLE
UNSCALABLE
BLOCKS CONTINOUS DEVELOPMENT
SLOW DEVELOPMENT
NOT FIT FOR COMPLEX APPLICATIONS

Figure 3: What is wrong with Monolith architecture

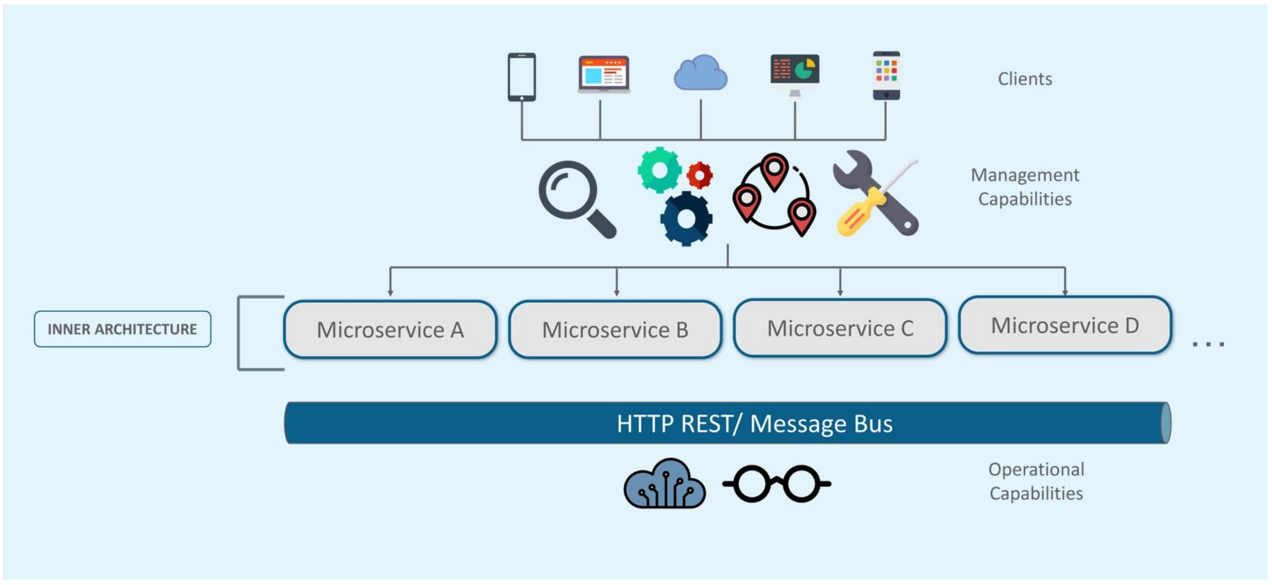


Figure 4: A basic microservice architecture

2.1.1.2 Process difference of software development

Conventional project management methods (waterfall, agile, etc.) typically lead to large groups/teams collaborating on just one monolithic implementation artefact. Project managers, engineers, and operating personnel may use such prototypes of differing degrees of accuracy, launching app candidates that could be tested by company, specifically when they obtain familiarity with a specific software and implementation stack. Nevertheless, there are also several problems with existing approaches that need to be addressed:

- Monolithic apps may become a "gelatinous blob of mud," in which neither one programmer (or community of developers) seems to have a complete understanding of the software.
- Only a small amount of recycle is possible through monolithic applications.
- Monolithic applications are notoriously difficult to scale.
- It's hard to sustain operating flexibility when deploying monolithic programme objects over and over again.

- Monolithic programmes are defined by the use of a specific development layer (for example, JEE or.NET), that may restrict the selection of "the best solution for said purpose."

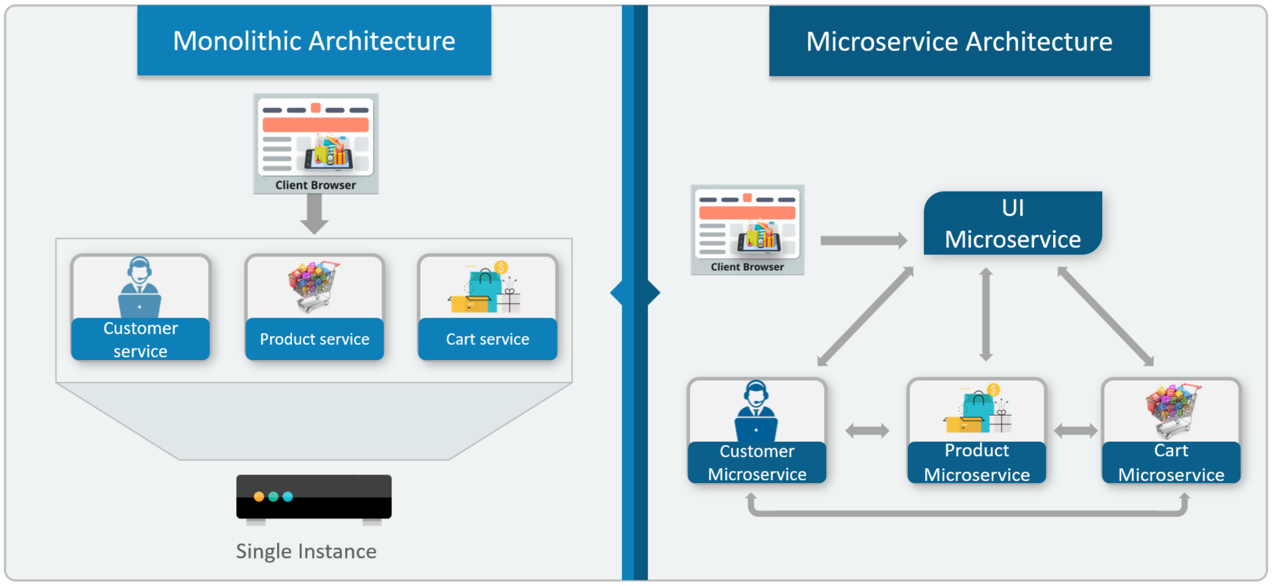


Figure 5: The differences between the two

When combined through public cloud architectures, API management ,automation technologies, a microservice offers a new way to design applications. A m onolith is dismantled into a range of standalone utilities that were already created, implemented, and operated independently. The below are some of the benefits:

Smaller services which are preferably designed by a limited community of developers, is welcomed. If the interfaces of microservices are presented using a standard protocolsuch as a REST API, they can be accessed and replicated by other apps and services with no need for language dependencies or same shared libraries.

Services should then be scaled individually of all other services when they are deployed as separate objects. Devs should use the best design process for tasks at hand when developing programmes separately.

The uncertainty that comes with this versatility comes at a cost. Managing a large number of connected services is difficult for many reasons:

- Working groups must be able to quickly identify resources that may be reused.
- Such programmes could provide manuals, testing systems, and other tools to make reusing them much simpler than starting from scratch. The interdependence of resources must be carefully supervised.

2.1.2 Microservices vs. monolithic design advantages

Microservices architecture over a monolithic architecture have major advantages. If fully implemented, a microservices-based model may have considerable economic advantages. This benefit could be reflected as a reduction in technological debts and also a significant improvement in efficiency. In typical DevOps, e.g technological debt out of a monolithic codebase is an observable fact. And separated modules of monolithic code use the very same storage and have access to the system itself. Although this can allow coding interfaces and implementing software kind of simpler, in the end, it eliminates the simplicity that ought to be part of the development growth process.

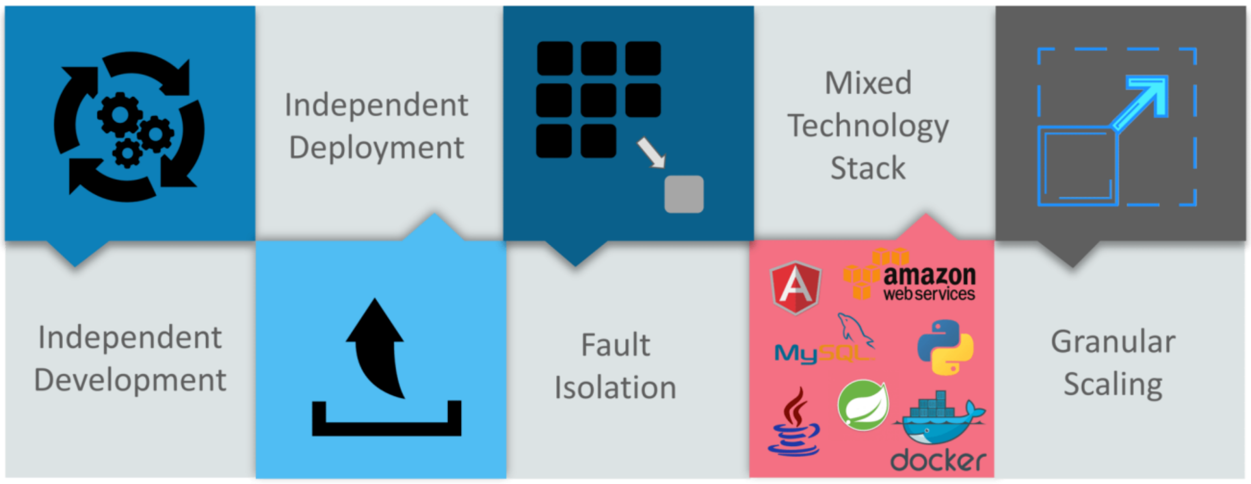


Figure 6: Advantages of Microservice architecture

Furthermore, a monolithic code base creates a degree of incompetence that raises technological leverage exponentially. Bug fixes, design improvements, capability additions, and many other updates to programmes, e.g have an effect on the programme overall, causing instability and developing an atmosphere where shortfalls may be implemented inadvertently.M onolithic code bases take longer to develop, are less flexible and adaptive, resultingly more difficult to manage, resulting in increased technological debt. A microservices-based architecture avoids many issues what monolithic architecture might cause like technical complexity, and thereby saves money in terms of the time and increased efficiency. Microservices have a significant advantage in terms of reducing technological debt but observable benefits don't really stop there.

Agility - DevOps will concentrate just on upgrading the specific types of applications by disintegrating features towards the most simple level and thereafter extracting the associated resources which eliminates the time-consuming economic integration that monolithic implementations are known for. Microservices shorten the production time, allowing it to be completed in weeks rather than months.

Efficiency - Using a microservices-based model will contribute in much more effective code and technology use. It isn't unprecedented to see substantial cost savings of up to 50% as when the amount of technology needed to operate a distributed objective is reduced.

Resilience - Spreading features throughout different services reduces the likelihood of a "single point of failure" in an appliance. consequently, systems will function faster, have less delays, and expand on requests.

Revenue - Quicker revisions and less downtime will boost sales. The incremental updates provided by microservices enhance customer satisfaction and interaction.

2.2 Version Control

Devs can use version control to keep a record of and handle updates to a software program's code. Version control becomes increasingly important as a software project progresses. Consider WordPress-It is a fairly large project at this stage. This wouldn't be secure or effective for a core programmer to modify the "official" source code if they were to focus on a new aspect of WordPress codebase. Rather, version control allows developers to securely divide and merge code. When a developer uses branching, they duplicate a portion of its code base (called the repository). The developer already can easily modify the section of the code without impacting the entire project.

Different instances of the same application are commonly installed in separate locations as companies plan, build, and deliver software, as well as the software's programmers are also operating on upgrades at about the same time. Computer bugs or additions are frequently only found in specific releases (because of the fixing of some problems and the introduction of others as the programme develops). As a result, being able to recover and execute various versions of the programme to decide which version(s) the problem resides in is critical for finding and repairing bugs.

It could also be possible to operate on two different versions of programme at the same time, for example, one variant with bug fixes but no new functionality (branch)and the other version with additional features. Developers might keep several backups of various iterations of the software and mark them accordingly at the most basic level. Many major tech programmes have used this basic technique. Although this approach can succeed, it is expensive since it requires the maintenance of several relatively-identical versions of the software. This necessitates a great deal of self-control on the parts of developers and frequently results in errors. Although the code base is same, it necessitates giving read-write-execute authorization to a community of designers this leads to just the burden of those handling permits to ensure that the code base isn't really corrupted, adding to the difficulty. As a result, programmes have been designed to simplify any or more of the revision management mechanism. This means that now the bulk of version control maintenance takes place behind the scenes.

Beneficial effects of using a version monitoring system include:

- The project production speed by fostering interaction;
- By improving coordination and support, the company will increase efficiency, speed up inventory development, and improve staff skills.
- Lower the risk of mistakes and clashes as the design is being developed by tracking any minor shift.
- Via this VCS, project employees, contributors may participate from anywhere across, regardless of the physical position.
- Assists in rehabilitation in the event of a crisis or unforeseen circumstance.
- It tells us Who, What, Where, and Why improvements were made.

GitHub is organisation that provides Git repository hosting in the cloud. It pretty much makes using Git for version control and sharing much simpler for individuals and organizations.

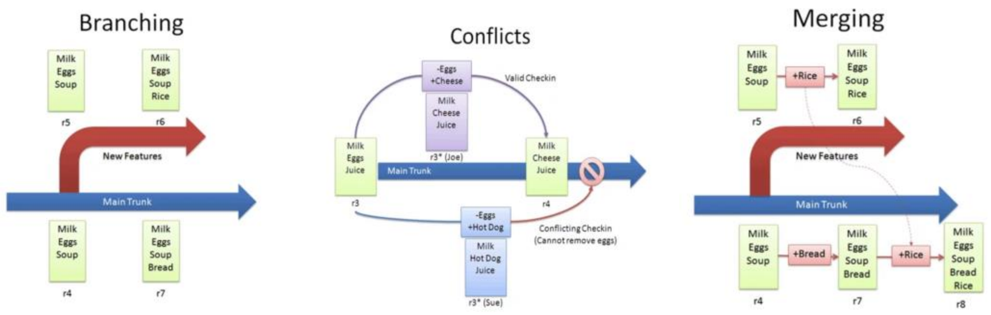


Figure 7: Issues that version controlling resolves and how [1]

2.2.1 Github

GitHub is simple user interface to provide functionalities of git version system in a simple fasion. Accessing Git neds a little more technological knowledge and terminal commands experience. However, since GitHub seems to be so user-friendly several people are using it to handle various kinds of tasks even for writing novels. Furthermore, anybody could register for free to maintain a community code repository, making GitHub especially successful for fully accessible projects.

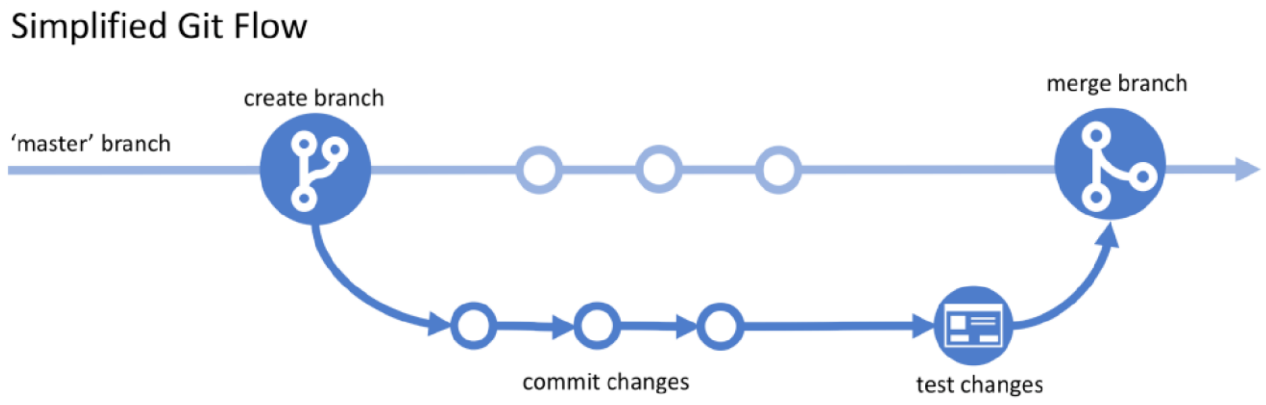
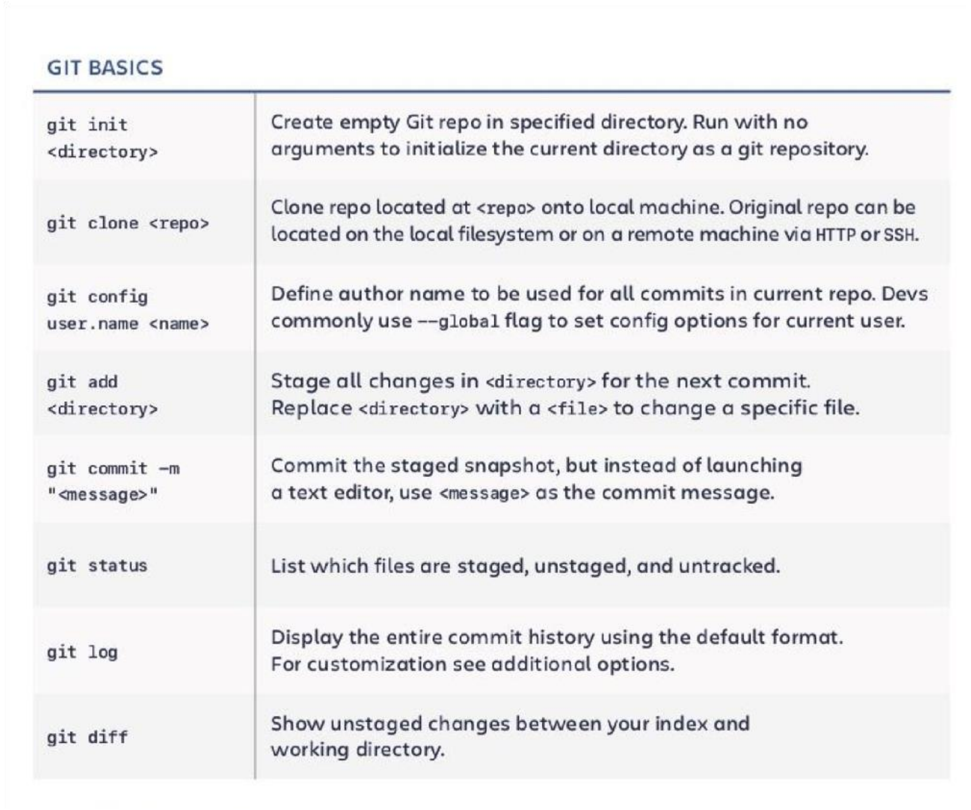


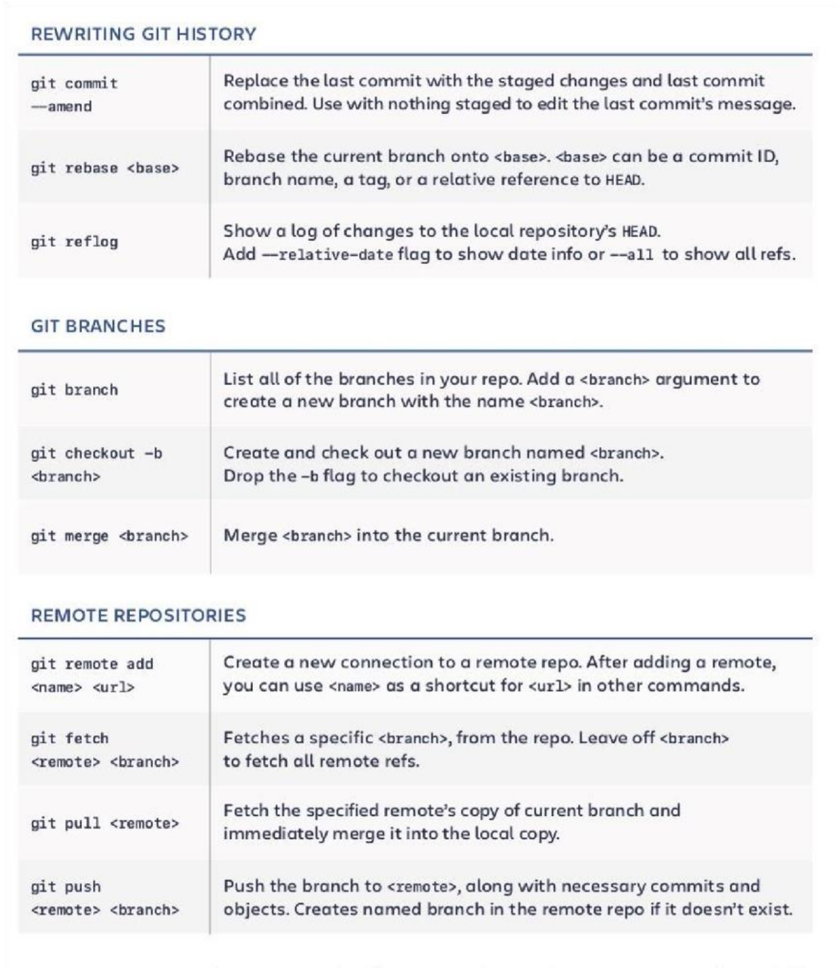
Figure 8: A simplistic git flow diagram



GIT BASICS

`git init <directory>`	Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository.
`git clone <repo>`	Clone repo located at `<repo>` onto local machine. Original repo can be located on the local filesystem or on a remote machine via HTTP or SSH.
`git config user.name <name>`	Define author name to be used for all commits in current repo. Devs commonly use `--global` flag to set config options for current user.
`git add <directory>`	Stage all changes in `<directory>` for the next commit. Replace `<directory>` with a `<file>` to change a specific file.
`git commit -m "<message>"`	Commit the staged snapshot, but instead of launching a text editor, use `<message>` as the commit message.
`git status`	List which files are staged, unstaged, and untracked.
`git log`	Display the entire commit history using the default format. For customization see additional options.
`git diff`	Show unstaged changes between your index and working directory.

Table 1 : Git basic commands



REWRITING GIT HISTORY

`git commit —amend`	Replace the last commit with the staged changes and last commit combined. Use with nothing staged to edit the last commit's message.
`git rebase <base>`	Rebase the current branch onto `<base>`. `<base>` can be a commit ID, branch name, a tag, or a relative reference to `HEAD`.
`git reflog`	Show a log of changes to the local repository's `HEAD`. Add `—relative-date` flag to show date info or `--all` to show all refs.

GIT BRANCHES

`git branch`	List all of the branches in your repo. Add a `<branch>` argument to create a new branch with the name `<branch>`.
`git checkout -b <branch>`	Create and check out a new branch named `<branch>`. Drop the `-b` flag to checkout an existing branch.
`git merge <branch>`	Merge `<branch>` into the current branch.

REMOTE REPOSITORIES

`git remote add <name> <url>`	Create a new connection to a remote repo. After adding a remote, you can use `<name>` as a shortcut for `<url>` in other commands.
`git fetch <remote> <branch>`	Fetches a specific `<branch>`, from the repo. Leave off `<branch>` to fetch all remote refs.
`git pull <remote>`	Fetch the specified remote's copy of current branch and immediately merge it into the local copy.
`git push <remote> <branch>`	Push the branch to `<remote>`, along with necessary commits and objects. Creates named branch in the remote repo if it doesn't exist.

Table 2: Git basic commands for branching, history and accessing remote repositories

2.3 Java to the Rescue

Java is a concurrent and object-oriented language of programming and a computing platform first published by Sun Microsystems in 1995. If you don't have Java enabled, you won't be able to use a wide range of applications and websites more are being developed day after day. Refusing to use Java is equivalent to refusing to use a technical infrastructure. It is praised and marketed because of its speed, stability, and dependability. Java is a programming language that was created specifically to be used in the Internet's distributed environment architecture. It can also be used to build full programmes which can run on a single device or be spread through a number of servers and clients. Java programmes are often compiled into "bytecode," which allows them to execute on any Java virtual machine (JVM) irrespective of the PC's hidden architecture. Java's language structure is similar to that of C and C++, in that it contains as few low-level functions as possible. One of the most important features of Java as a programming language is its automated board memory, often known as garbage specialist. When articles are created, the author decides, and the Java runtime is in charge of recovering memory until the items are no longer needed.

What makes Java a really prominent programming language amongst today's software developers?

The reason is largely due to Java's long history of comprehensive testing, upgrading, and distribution continuity. A devoted group of Software engineers, architects, and pioneers have evaluated, enhanced, expanded, and validated Java. It has grown steadily over the years despite and it's almost two-decade roots. It is developed to enable the development of lightweight, high-performance software for the broadest possible variety of computing technologies, thereby facilitating the fundamental aspects of holistic availability and cross-platform engagement. Java has gained considerable interest among developers because it allows them to:

- Write applications on one computer and run it on almost every another.
- Make java programs that run in a web browser, link to accessible online services.
- Create server-side software for web forums, shops, elections, and the rendering of HTML forms among other things.

- Using the Java programming language, you may integrate services and applications to build highly specialised services and applications.
- Create apps for cell phones, remote processors, microcontrollers, wireless modules, cameras, gateways, consumer goods, and almost every other computer system.

2.3.1 JDBC

A standard that specifies a basic abstraction (API or Protocol) for java Software applications to connect with different databases. It implements the Java database functionality standard in the language. This is used to create applications that link to databases. Databases and spreadsheets can be accessed using JDBC as well as the database driver. JDBC APIs can be used to view business data contained in a relational database (RDB). JDBC is an API (application programming interface) for interacting with databases in Java programming. JDBC classes and interfaces enable applications to submit user requests to a given database.

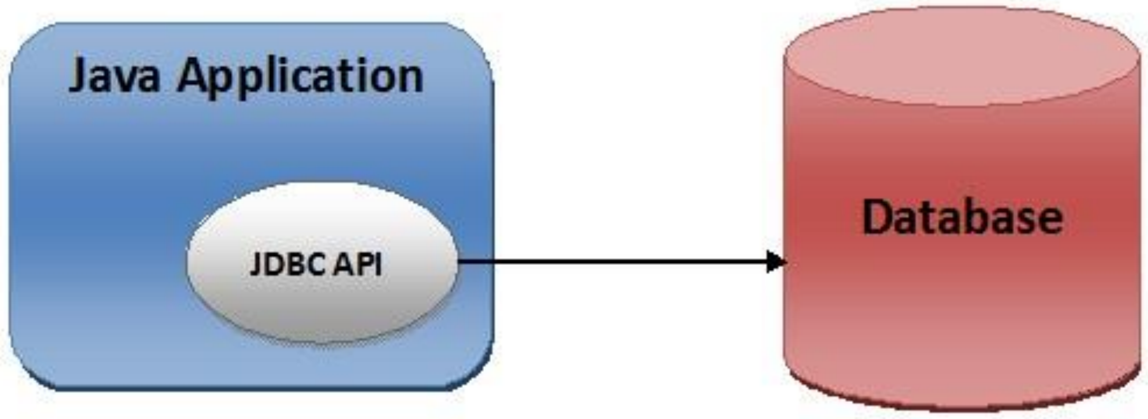


Figure 9: JDBC

The JDBC API's primary artefacts are:

- To create relations, a DataSource object is used. Driver Manager is used to create a link, it is recommended that you use a DataSource item.
- A Connection reference manages the database connection. By triggering the methods associated with this object, an implementation may change the behaviour of a relation. The relation object is used by an application to construct claims.
- SQL statements are executed using the Argument, PreparedStatement, and CallableStatement elements. When an application intends to reuse a declaration several times, it uses a PreparedStatement object.
- The programme sets up the SQL it can use. The programme will then set values for variables in the prepared SQL statement until it has been prepared. The assertion can be run several times, each time with a different set of parameter values. The CallableStatement includes ways of extracting the retained procedure's return values.
- The results of a query are stored in a ResultSet object. When a statement object executes a SQL query, it returns a ResultSet to the server. The
  ResultSet object contains methods for iterating over the query's results.

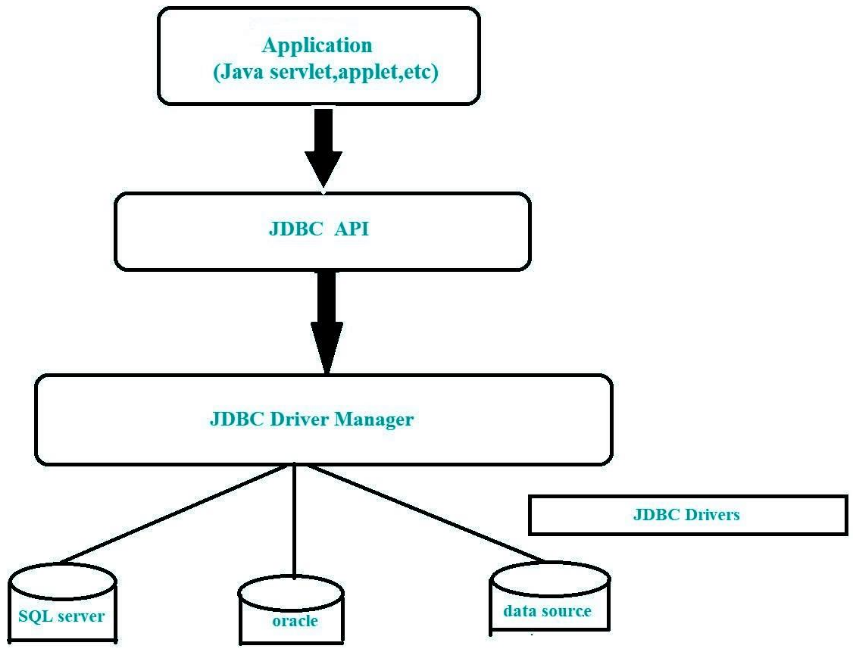
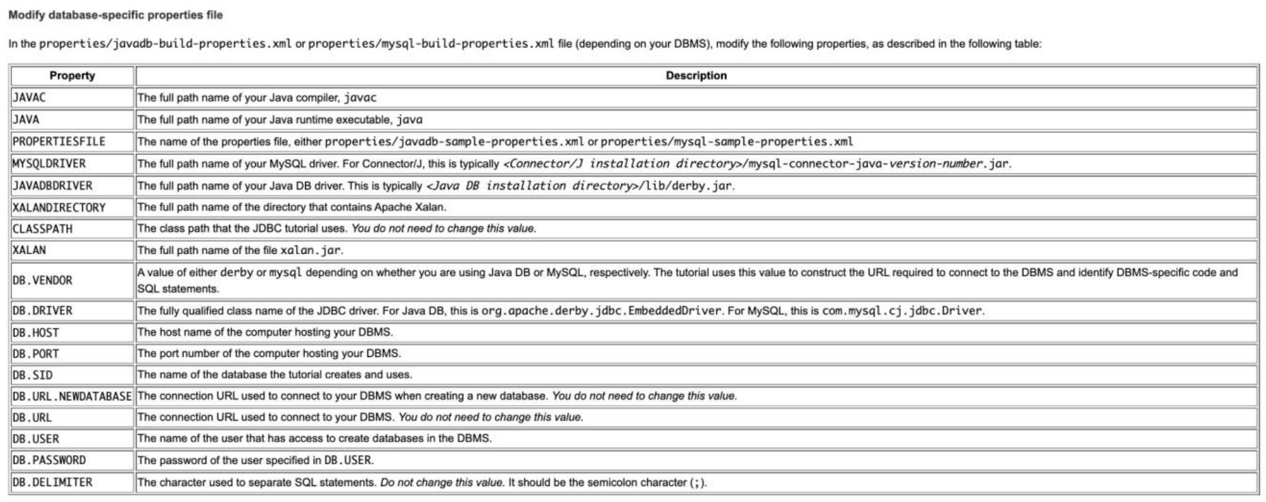


Figure 10: JDBC architecture



Modify database-specific properties file

In the `properties/javadb-build-properties.xml` or `properties/mysql-build-properties.xml` file (depending on your DBMS), modify the following properties, as described in the following table:

Property	Description
`JAVAC`	The full path name of your Java compiler, `javac`
`JAVA`	The full path name of your Java runtime executable, `java`
`PROPERTIESFILE`	The name of the properties file, either `properties/javadb-sample-properties.xml` or `properties/mysql-sample-properties.xml`
`MYSQLDRIVER`	The full path name of your MySQL driver. For Connector/J, this is typically *`<Connector/J installation directory>`*`/mysql-connector-java-`*`version-number`*`.jar`.
`JAVADBDRIVER`	The full path name of your Java DB driver. This is typically *`<Java DB installation directory>`*`/lib/derby.jar`.
`XALANDIRECTORY`	The full path name of the directory that contains Apache Xalan.
`CLASSPATH`	The class path that the JDBC tutorial uses. *You do not need to change this value.*
`XALAN`	The full path name of the file `xalan.jar`.
`DB.VENDOR`	A value of either `derby` or `mysql` depending on whether you are using Java DB or MySQL, respectively. The tutorial uses this value to construct the URL required to connect to the DBMS and identify DBMS-specific code and SQL statements.
`DB.DRIVER`	The fully qualified class name of the JDBC driver. For Java DB, this is `org.apache.derby.jdbc.EmbeddedDriver`. For MySQL, this is `com.mysql.cj.jdbc.Driver`.
`DB.HOST`	The host name of the computer hosting your DBMS.
`DB.PORT`	The port number of the computer hosting your DBMS.
`DB.SID`	The name of the database the tutorial creates and uses.
`DB.URL.NEWDATABASE`	The connection URL used to connect to your DBMS when creating a new database. *You do not need to change this value.*
`DB.URL`	The connection URL used to connect to your DBMS. *You do not need to change this value.*
`DB.USER`	The name of the user that has access to create databases in the DBMS.
`DB.PASSWORD`	The password of the user specified in `DB.USER`.
`DB.DELIMITER`	The character used to separate SQL statements. *Do not change this value.* It should be the semicolon character (`;`).

Figure 11: Modifications for using database with JDBC

2.4 Docker

A development framework intended for developing methods dependent on containers, which are lightweight and compact operation environments that share the operating system core but operate in autonomy. Though containers as a term have been here for a long time a free and open - source initiative called Docker released in 2013 helped in popularising the platform and propelled cloud-native architecture movement of containerization and microservices in software engineering [3].

It is built on a client-server model. Docker client communicates with the Docker daemon, that handles the construction, execution, and distribution of the Docker containers. You can execute the Docker client and daemon within the same machine, or you could just bind a Docker client to a Docker daemon that is located elsewhere.

The REST API, UNIX socketsor a network adapter are used by the Docker client and daemon to connect. Docker Compose is yet another Docker client which allows one to deal with apps made up of several containers. For example, developers write code individually and then use Docker containers to sync it with teammates. They employ Docker to deploy their software and run computerised testing in a testing environment.

Developers should correct vulnerabilities in the implementation environment before deploying them to the test environment for testing and evaluation. When the research is over, it's only a matter of pushing the modified picture to the manufacturing environment to bring the patch to the consumer.

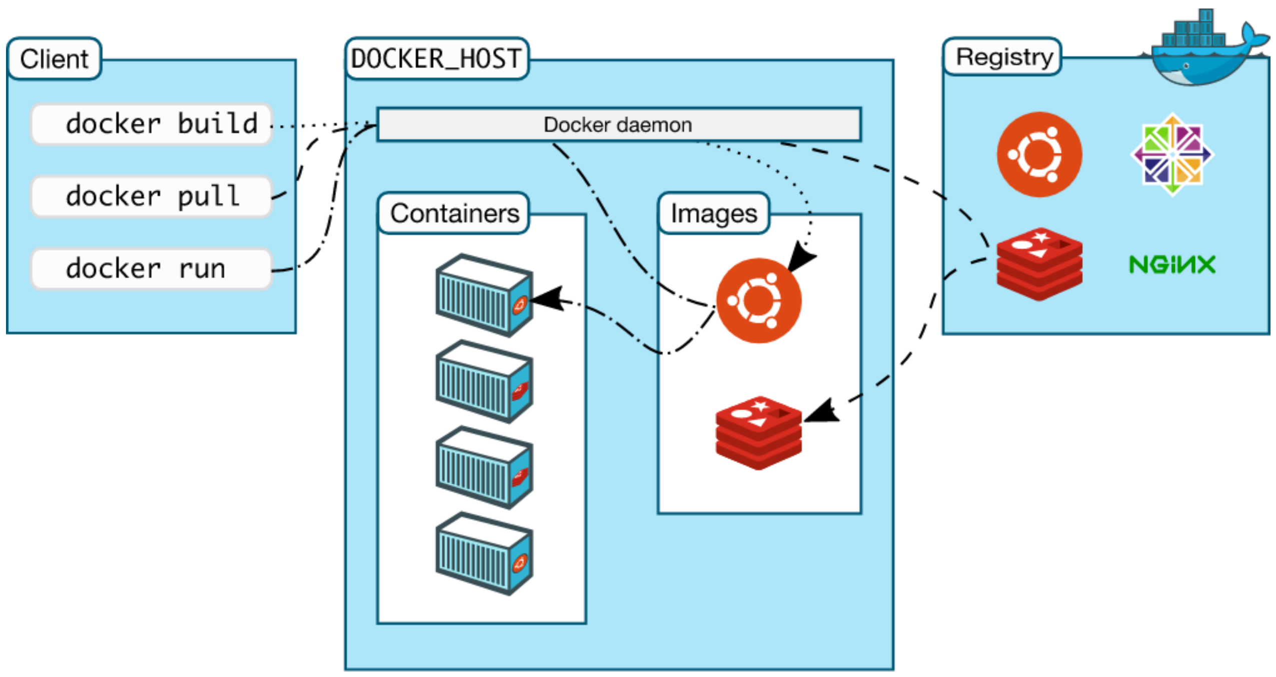
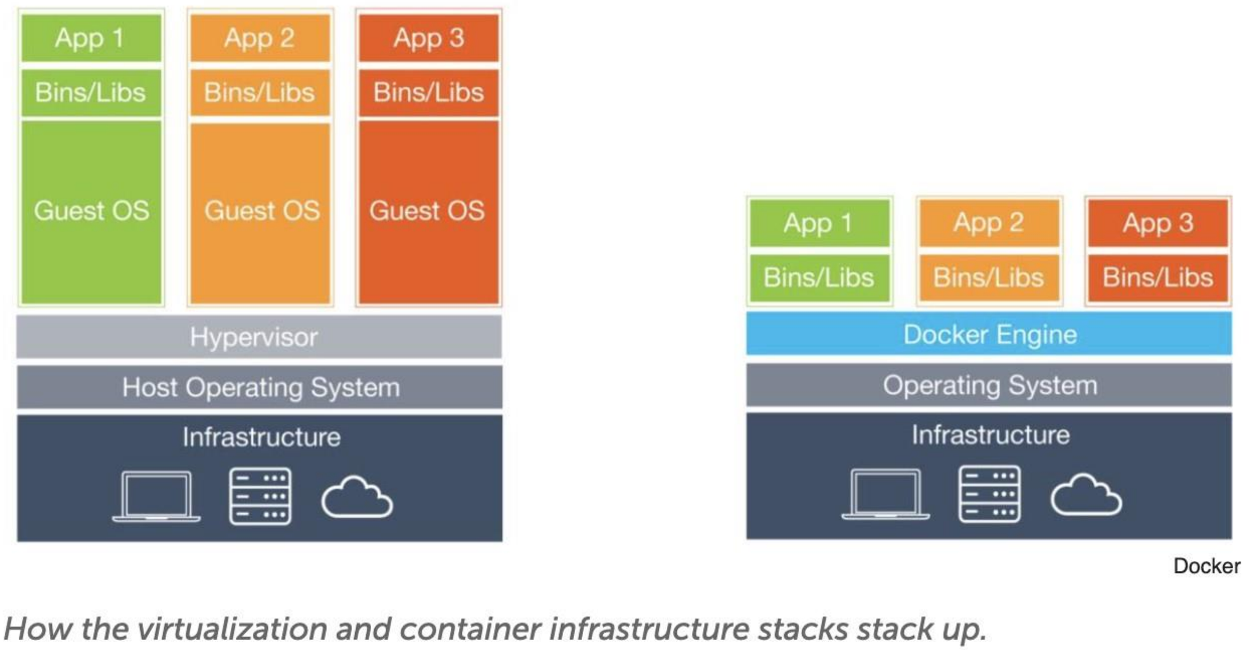


Figure 12 : Docker explained [3]

Aim of advanced software architecture is to maintain programmes running on the very same host or network separated from each other and so that they don't compete with each other's activity or maintenance. Due to the packages, libraries, and other programme components needed for them to run, this can be challenging [3]. Digital machines are one solution to this challenge, since they hold programmes on the same hardware completely apart, reducing conflicts between software modules and rivalry for hardware resources to a minimum. Virtual computers, on the other hand, are large (each needs its own operating system, so they're usually gigabytes in size) and difficult to manage and update. On the other hand containers separate the execution environments of programmes while sharing the corresponding OS kernel. They're usually calculated in megabytes they use a fraction of the power of virtual machines, and boot up very quickly. They can be stacked far more together on the same hardware and turned up and down in mass with much less commitment and overhead.

Containers offer an extremely effective and fine grained method for integrating software components into the types of programme and utility stacks required in today's enterprise, as well as for maintaining such software components optimized and managed. Docker simplifies the project development by encouraging developers to operate in simplified settings using localized containers to deliver the apps and services. Continuous integration and continuous distribution (CI/CD) workflows benefit greatly from containers.



2.5 Spring Boot

“Spring boot came into existence from late 90s and became popular in the last decade as it supports the development of microservices. The main reasons why it is used are as follows -

1. Applications build from are production ready and ready to be deployed.
2. Many features are also provided to the users like metrics, health checks, etc.
3. Code automation is there.

4. There is no need for XML configuration which was first needed in older version.
5. Lastly, Configuration which is to be done by the developers in the other frameworks, it is not needed now, all of it is automated. All functions like third party API handling and import of libraries in the module are automated inside this framework."

There are many projects which comes under this which can be used and they all are starter projects, including dependencies in application.properties. They are turned out to be very useful for the developers as they do not have to write the code for setup of projects and also in developing microservice applications. Examples of the starter projects are -

1. Spring-boot-starter-web
2. Spring-boot-starter-test
3. Spring-boot-starter-jpa

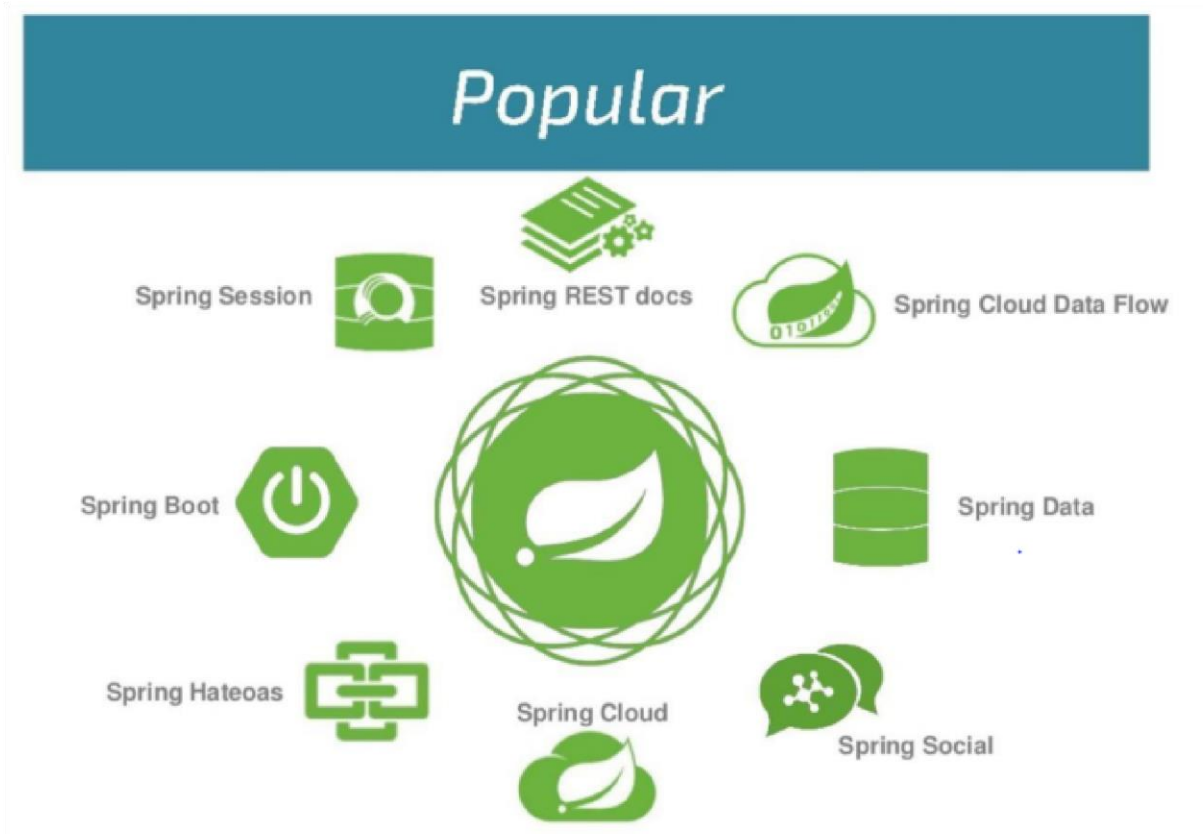


Figure : Popular spring projects

2.6 Spring MVC

A software design pattern that divides the following device or subsystem components:

- **Model -** Information about the application's or its components' current state. Modification and entry procedures are possible. The model seems to be how we implement rules on data to identify the principles that our framework handles. All in a software application is modelled as data that could be conveniently managed. What then is the difference between a person, a book, and a message in an app? There isn't anything to it; it's just data that has to be interpreted including a set of laws. For example, the date must be older than the current date, the mail must always be formatted correctly, the name should be longer than "x" characters, and so on. When a user approaches the controller with a message, the controller approaches the relevant model that produces a data representation of customer's query.

- **View -** A point of view is a way of looking at something (model). This isn't confined to a graphic representation audio or derived data may also be used. A single model may also have different views. User interacts with a view. The information obtained from model data is used to construct views. A view asks the model for input so as to re-present the result to the user. The data via chats graphs, and tables can also be seen in the view. Any customer view, e.g would contain all UI elements such as text boxes, drop downs, and so on.

- **Control -** Deals with external feedback to the system that causes the model to change. It's possible that the control and the view are connected (in the case of a UI). Other external input (such as network commands) will, however, be processed regardless of the vision. The controller acts as the application's personal assistant, coordinating the model and display in order to fulfil an user's query. When customers click on some GUI feature to execute an operation, the customer's signal is approved as an HTTP get,post or any other request. The main responsibility of a controller is to communicate with model to manage the retrieval of just about any resources needed to operate. The controller essentially tells the appropriate model for the mission at hand when it receives a user order. Put simply the model is the data for programme. The information is "modelled" in such a way that it is simple to store, access, and modify.

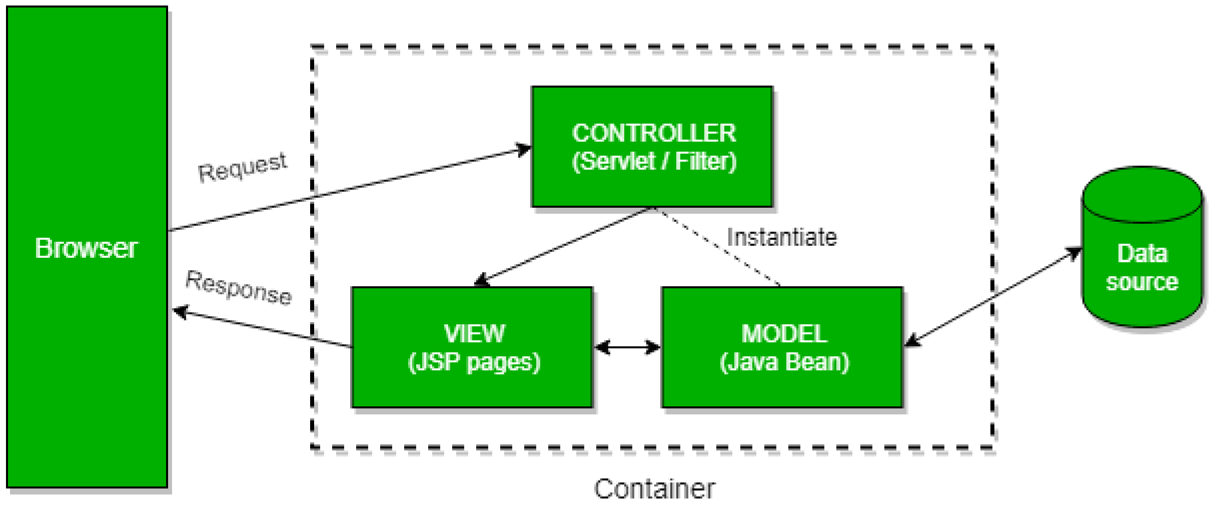


Figure 12: How MVC components can be viewed

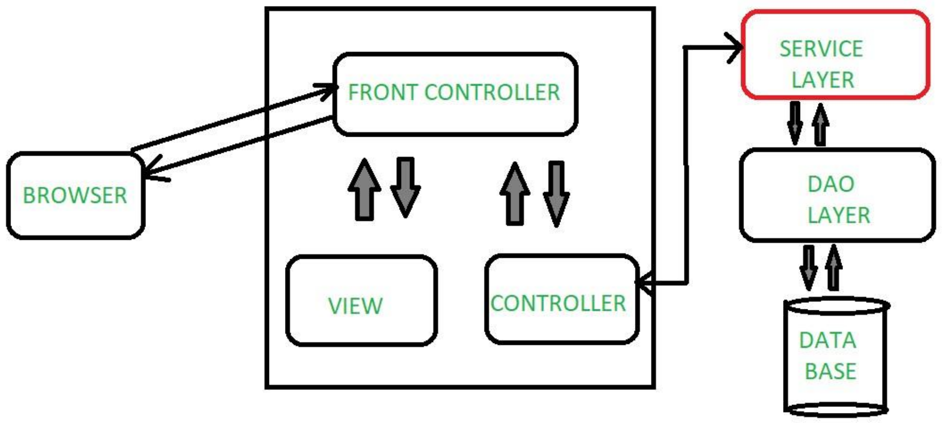


Figure : MVC with service layer

Advantages -

- Separate responsibilities - Spring MVC distinguishes each function, with a specialised object fulfilling the model object, controller, command object, display resolver, DispatcherServlet, validator, and so on.
- It develops and deploys the programme using a light-weight servlet container.
- It requires simple referencing across domains, such as from site controllers to business objects and validators, and offers a stable configuration for both platform and application types.
- Rapid development - The Spring MVC framework allows for rapid and concurrent development.
- Reusable business code enables one to reuse current business artefacts rather than making new ones.
- Test is easy - In most cases, we build JavaBeans classes in Spring that allow inserting test data via setter methods.
- Mapping is flexible - Spring offers specific annotations to enable the page to be quickly redirected if need comes.

While MVC in itself is a great design pattern, sometimes a layer other than the three called as Service layer can be added to make it MVCS model. Why do we need a Service layer when we can operate on data with Model itself? In general, the DAO layer must be a light , with its main purpose being to get a link to database, which is often abstracted so that separate database backends can be used around each other. The service layer's aim would be to provide logic for processing data sent from and to the DAO and client. These two components are sometimes combined into the same module, and sometimes into the same code, but they are still seen as separate logical entities. So the answer to the question is -

- Service layer ensures code modularity; business logic and guidelines are defined in the service layer which further calls the DAO layer. Only responsibility for DAO layer is to communicate with the database.
- Provides Security - If there is a service layer with no connection to the database, gaining access to the database from the customer rather than via the service becomes much more challenging. Any intruder who has taken control of the client won't be able to access the info if database can't be reached directly from the client (and there's no simple DAO component serving as the service).
- Serves Loose Coupling - This may also be included in the program to provide loose coupling. Let us just say a controller had 50 methods and it calls 20 Dao methods. At some stage in the future plan to modify the Dao methods that serve these controllers is made. Every 50 method in the controller must be modified. Rather, if there are 20 service methods that call those particular 20 Dao methods only 20 of them are required to be modified to refer toward a new Dao.

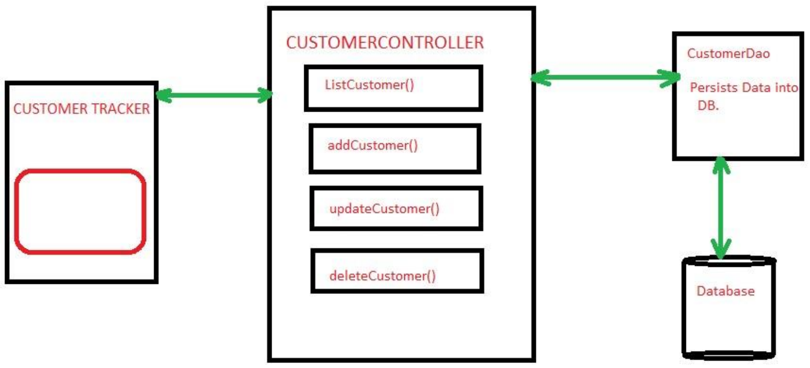


Figure 13: Application flow diagram without service layer

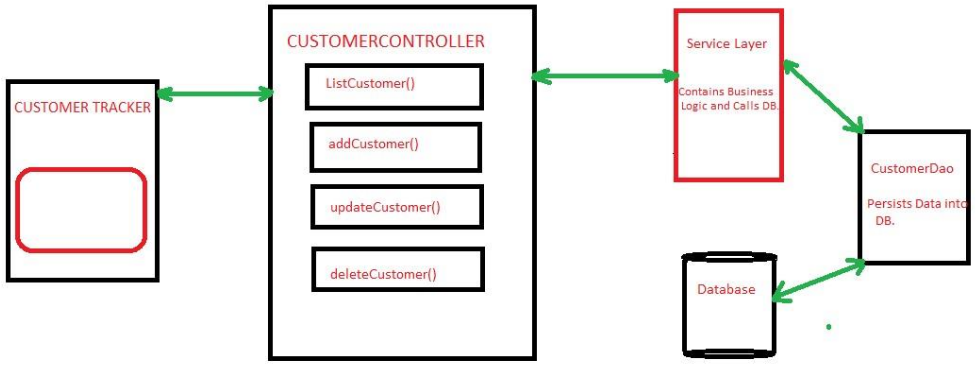


Figure 14: Application flow diagram with Service layer

2.7 REST API

API stands for Application Programming Interface, and that it is a compilation of communication protocols and subroutines that allow different programmes to communicate with one another. A developer would use a variety of API technologies to keep application simpler and quicker. An API allows developers to create their software programmes in a more effective manner.

An API facilitates communication between two programmes or software by supplying them with the resources and functionality they provide in straightforward terms. It accepts the participant's request and sends it over to the service provider, after which it delivers the service provider's response to the desired user.

REST is a mechanism that makes use of known protocols. Although REST can be used for virtually any protocol, it is most commonly used with HTTP for Web APIs. This suggests that in order to use a REST API architecture, programmers shouldn't need to install any libraries or extra applications. It's noteworthy for its incredibly flexible coating. REST can manage various kinds of calls, produce different formats of data, and also modify architecturally with the proper application of hypermedia when data is not bound to mechanisms and services.

What exactly is REST(Representational State Transfer) and what does it imply? Text representations are transferred, accessed, and manipulated in a stateless fashion. It offers consistent interoperability between different applications on the internet when properly implemented. The word "stateless" is important because it enables systems to interact with each other regardless of their state. A Uniform Resource Locator (URL) is used to access a RESTful API function (URL). This rational naming distinguishes the resource's identity from what is acknowledged or retrieved.

2.8 Mockito Testing FrameWork

Developer side testing is highly vital for any application to obtain better results, and today every reputable business company is doing it. It is not the task of a Quality Assurance engineer; it is solely the responsibility of a Software engineer, and Mockito is one of the most popular frameworks for it. It mocks the object as well as the method that has to be tested. It is built on Java and makes use of annotations. It involves creating false objects that act as though they are calling the real function, and then comparing the actual result to the expected result supplied by the user. It's a method of testing code coverage to determine how much of our code is covered before releasing it, as well as to determine whether any code is included that isn't helpful. In it, a false object is created with the same information as the real item, and the user compares the outcome to the result supplied.

Mockito unit testing is completed by mocking the conditions for the classes under test, followed by code execution. Approve the result after the code has been performed to see if the taunting class restores a comparable reaction true to form or not.

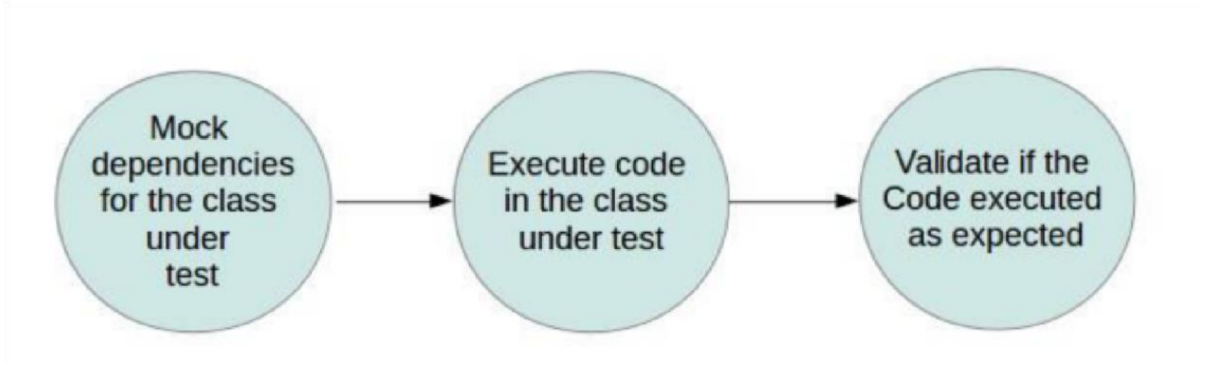

Mock dependencies for the class under test
Execute code in the class under test
Validate if the Code executed as expected

Figure 15: Unit testing using Mockito

2.9 Functional Requirements

2.9.1 Authentication

- Depending on the role of the user different screens should be shown.
- Login onto the server is mandatory for the user before processing

any information further.

- Tokenization is used for authentication whenever any API call is made from frontend to backend.
- This process generates a token which is dependent on the type of user logged hitting any API.

2.9.2 User experience and interface

- Compatible with different browsers.
- Follows section 508.
- Regular updates of the platform.
- Full review for the user and error handling so user can understand of the issue.
- User friendly interface.
- Beta version of UI checked with many firms.
- Customisable UI for business firms.

2.9.3 Performance

- Caching is available for the user.
- To make the size of application not too heavy, optimisation over the image is also implemented
- Management of Cookie
- To make the application up and running for all time and to protect it from lagging, number of HTTP requests are reduced.
- Taking care of user information by encrypting it.

2.10 Non-Functional Requirements

2.10.1 Self Service

Many individuals used to have to travel to any official premises and wait a long time for their application to be moved from one department to another, but this application is a self-servicing platform that does not require any interaction with agents or workers; just the validation is done by the company.

2.10.2 Responsiveness

Provides the best user experience and this is because of fast response and it provides equal priority to every user

2.10.3 Accessibility

It provides maximum accessibility to its users and with this comes with the use of minimum technical knowledge, this is accessible to every age member with ease.

2.11 Technological Approach

Specifications - Project I worked on utilised a variety of technologies, the implementation of which takes less time and is simple; further, it ensures that the technology's full potential is utilised. The following technologies are utilised for the backend I was assigned:

Backend -

1. Java
2. Spring boot

3. PostgreSQL
4. REST API
5. Docker

Tools used -

1. IntelliJ
2. Swagger
3. Postman
4. Linux
5. Github

2.12 Unit Testing

Unit testing is one of the most important activities that a developer should adopt and practise; he or she should make it a rule that if the developer writes a piece of code, he or she should also have his or her unit test code ready; this not only benefits the user by allowing them to test it, but it also helps the developer if he or she decides to include any code, making it easier for him or her to enforce his or her code.
Unit testing examines the code, and one of its most important properties is coverage. The primary goal of unit testing is to ensure that each piece of code functions as intended.
When a large function is written in a modular manner, each piece of code/method in which it is broken is referred to as a unit.

It is very important as developer sees the proper execution of code and helps him to debug the code. Tools used to unit test are -

1. Jtest
2. Junit
3. Mockito

2.13 Testing APIs with Postman –

- ❑ Either write the API manually or Import it in any format like text file, folder, link or raw text of curl.
- ❑ If the API is of POST method then write request body in the provided column, and run the API.
- ❑ Authorisation can also be provided either by Bearer token of some credentials.
- ❑ “Headers can be given, by this postman acts as the browser running the API, the difference is that it does not shows the html page as the result instead shows the response data.”
- ❑ Postman supports multiple request methods like GET, POST, PUT, DELETE etc.
- ❑ The response can be checked in the response column.

2.14 Code Coverage

In Portal, testing of every controller and service including all common helper function is done and the code coverage of 95.7% came out of 100%. Whenever any new piece of code is added then it can be tested immediately and precise areas of concern cab be recognized by the developer.

2.15 COMMON ERRORS:

Below is a list of errors that are frequently encountered while configuring a journey. It contains a list of possible reasons that cause that error.

It may happen that the error occurs due to a reason not in the list. We will keep updating as we come across more.

You can check these under Inspect : Network & Console

1. 500 : Something went wrong

    Possible Reasons :

    1. Error in Script task format
    2. Field used in script task (Get Variable) or Service task is not defined before the tasks are triggered
    3. Error in service task format
    4. Error in function/API that was called in service task
    5. The ids used in script task or service task can be different compared to the field ids used in forms
    6. Server related error
    7. API error in case the field has an API error

8. Mapping error in case an API has been triggered 9. Assignee or Task Id not present in the User task
10. Assignee is incorrect.
11. Mapping of variables in a Decision Table
12. Boolean values such as TRUE/FALSE are defined as string
13. If the field is rest-input, make sure to add type : rest in the meta

2. Missing field error

Possible Reasons :

1. A field called as child to another field has not been defined. Ex : In case of an accordion,table or display-of-selection

3. 401

Possible Reasons :

1. Invalid token. The token could have expired
2. Invalid user credentials. Either credentials entered are incorrect or have not been defined

4. TypeError: Cannot read property 'map' of null : This is mostly due to error in javascript

Possible Reasons :

1. Script task error
2. Error in meta . For example the component dragged is text but form_type is dropdown and no enums and options have been defined for the dropdown
3. The field expects an array but value is being saved in a different format

Although [3] states the kinds of errors we should narrow down our error codes to. Status codes for your APIs when it comes to the relationship between an app and an API, there are only three possible outcomes:

- All went well, so it was a success.
- The application made a mistake – client fault
- Something went wrong with the API – server error

Generally 200 is OK, 400 is for Bad Request and 500 denotes Internal Server Error.

But [3] says if you’re If you're not comfortable reducing all your error conditions to these 3, try picking among these additional 5:

- 201 - Created
- 304 - Not Modified
- 404 – Not Found
- 401 - Unauthorized
- 403 - Forbidden

CHAPTER 3: Conclusions

3.1 Findings

During the creation of this project, I learned a variety of skills, including how to create clean code, how to build a database, many phases of web development, backend development, system design concepts, Java, Spring framework, SQL, and Rest APIs. Today's world is making enormous strides in web development, and now is the time to act. Many large corporations are concentrating their efforts only on this sector. Microservice architecture is superior to monolithic architecture in terms of functionality and speed. This is because when a monolithic design architecture's server fails, the entire product suffers, however this is not the case with microservice architecture.

3.2 Conclusion and Future Scope

My role at ZopSmart is that of a backend developer, where I enhance functionality of the existing websites for ecommerce and to make sites perform as they should, to validate data is being passed correctly or fixing bugs whenever one is encountered. I work on applications created in favour of customers' experience, part of the huge projects, offers given to them, updating, collecting and creating their data in the database for different stores. This procedure significantly reduces the number of mistakes because everything is computerized and requires very little human participation and value towards advancement of use of technology in daily lives for the better. In future, as the number of customers increase, we'll be needing strategies to make like querying from databases faster and easier, to perform effective load balancing, to reduce time and computational costs without affecting performance, etc. I intend to learn and develop more efficient/scalable solutions for enhancing the technological trade and commerce, and making humans' experience with my company's technology better and more efficient.

References

[1] https://betterexplained.com/articles/a-visual-guide-to-version-control/

[2] https://docs.docker.com/get-started/overview/

[3] https://pages.apigee.com/rs/apigee/images/api-design-ebook-2012-03.pdf