

Distribution Management System

A Project-II Report

Submitted in partial fulfillment of requirement of the

Degree of

**BACHELOR OF TECHNOLOGY in COMPUTER
SCIENCE & ENGINEERING**

BY

**NILESH KESHWANI
EN18CS301159**

Under the Guidance of

**Dr. Kailash Chandra Bandhu
Mr. Binod K Mishra**



**Department of Computer Science & Engineering
Faculty of Engineering
MEDI-CAPS UNIVERSITY, INDORE- 453331**

JAN-MAY 2022

Distribution Management System

A Project-II Report

Submitted in partial fulfillment of requirement of the

Degree of

**BACHELOR OF TECHNOLOGY in COMPUTER
SCIENCE & ENGINEERING**

BY

**NILESH KESHWANI
EN18CS301159**

Under the Guidance of

Dr. Kailash Chandra Bandhu

Mr. Binod K Mishra



Department of Computer Science & Engineering

Faculty of Engineering

MEDI-CAPS UNIVERSITY, INDORE- 453331

JAN-MAY 2022

Report Approval

The project work “**Distribution Management System**” is hereby approved as a creditable study of an engineering application subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite for the Degree for which it has been submitted.

It is to be understood that by this approval the undersigned do not endorse or approved any statement made, opinion expressed, or conclusion drawn there in; but approve the “Project Report” only for the purpose for which it has been submitted.

Internal Examiner

Name:

Designation

Affiliation

External Examiner

Name:

Designation

Affiliation

Declaration

I hereby declare that the project entitled “**Distribution Management System**” submitted in partial fulfillment for the award of the degree of Bachelor of Technology in ‘Computer Science & Engineering’ completed under the supervision of **Dr. Kailash Chandra Bandhu, Associate Professor**, Faculty of Engineering, Medi-Caps University Indore is an authentic work.

Further, I declare that the content of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for the award of any degree or diploma.

Signature and Name of the Student with Date

Certificate

We, **Dr. Kailash Chandra Bandhu and Mr. Ramesh Pallikonda (External Guide)** certify that the project entitled “**Distribution Management System**” submitted in partial fulfillment for the award of the degree of Bachelor of Technology by **Nilesh Keshwani** is the record carried out by him under our guidance and that the work has not formed the basis of award of any other degree elsewhere.

Dr. Kailash Chandra Bandhu
Computer Science & Engineering
Medi-Caps University, Indore

Mr. Ramesh Pallikonda
Senior Software Engineer
IMPETUS Technologies India Pvt. Ltd.

Dr. Pramod S. Nair
Head of the Department
Computer Science & Engineering
Medi-Caps University, Indore

Offer Letter of the Project work-II/Internship

IMPETUS

Date: 21-Dec-2021

To,
Nilesh Keshwani
House No. 373 Sector B Suryadev Nagar
Near Madan Mast Sweets, Hawa Bunglow - 452009

Dear Nilesh Keshwani,

Pursuant to our discussions, **Impetus Technologies India Pvt. Ltd.** is pleased to make you an offer of employment as **Project Trainee** at grade **X** starting on **18-Jan-2022**.
Your Stipend will be INR **10000/-** per month.

All terms and conditions of your engagement will be in accordance with this letter read along with (i) Annexure A to this letter, (ii) the Appointment Letter (to be issued to you on successful completion of your Training and fulfillment of all documentation requirement), (iii) the Non-Disclosure Agreement. Formats / copies of the above-mentioned documents are available with us and you are required to read and understand the same prior to your acceptance of our offer for training engagement. In addition, you shall also be required to abide by all the policies and procedures of the Company including those provided on the Company's intranet.

You will be eligible to join as **ASE** at Grade **G4** only after successful completion of your Post-graduate degree and on submission of the pass certificate, your CTC will be revised to **500000**.

Please note that this offer is conditional upon satisfactory feedback from your references and necessary background, academic, medical, credit/financial and criminal checks. Our offer is also contingent upon your full and complete disclosure to the Company of any and all agreements (non-competition, non-solicitation, employment, confidentiality or otherwise) with any prior employer, clients, principals, partners or others which in any way limit you either contractually or otherwise from engaging in any business activities required or contemplated by the Company in this offer of training engagement. The Company reserves the right to withdraw this offer of training engagement (for training) without any obligation whatsoever, if it determines or believes that any contractual or other obligation may materially limit your ability to engage in business activities for the Company.

The Company reserves the right to withdraw this offer of employment without any obligation whatsoever, in the event that it presumes or suspects or determines or believes that any commercial or

Impetus Technologies India Pvt Ltd

CIN: U72100MP2000PTC014455

Impetus IT SEZ Campus, Survey no. 291, Badiya Keema, Bicholi Mardana-Ambamoliya Road, Indore-452016 (M.P.) India., Phone: +91.731.4743600. Fax: +91.731.473611 Regd. Office : Sarda House, 24-B, Palasia, A.B. Road, Indore-452001 (M.P.) India., Phone: +91.731.4269300, Fax: +91.731.4071256

Completion certificate/Letter

IMPETUS

Certificate from the Company

To whomsoever it may concern

This is to certify that Mr. Nilesh Keshwani has been allowed to carry out a project titled Case Study (Distribution Management System) in our organization.

The duration of the project will be from 18-01-2022 to 18-05-2022.

Signature



Impetus Technologies India Private Limited
(Formerly known as Impetus Infotech (India) Private Limited)
CIN - U72100MP2000PTC014455

Regd. Office : Sarda House, 24-B, Palasia, A.B. Road, Indore-452001 (M.P.) India.
Phone: +91.731.4269300, Fax: +91.731.4071256.

Acknowledgements

I would like to express my deepest gratitude to Honorable Chancellor, **Shri R C Mittal**, who has provided me with every facility to successfully carry out this project, and my profound indebtedness to **Prof. (Dr.) Dilip K Patnayak**, Vice Chancellor, Medi-Caps University, whose unfailing support and enthusiasm has always boosted up my morale. I also thank **Prof. (Dr.) D K Panda**, Pro Vice Chancellor, **Dr. Suresh Jain**, Dean Faculty of Engineering, Medi-Caps University, for giving me a chance to work on this project. I would also like to thank my Head of the Department **Dr. Pramod S. Nair** for his continuous encouragement for betterment of the project.

I express my heartfelt gratitude to my **External Guide, Mr. Ramesh Pallikonda**, Senior Software Engineer, Impetus Technologies India Pvt. Ltd as well as to my Internal Guides, **Dr. Kailash Chandra Bandhu**, Associate Professor and **Mr. Binod K Mishra**, Assistant Professor, Department of Computer Science & Engineering, Medi Caps University, without whose continuous help and support, this project would ever have reached to the completion.

I would also like to thank to my team at Impetus Technologies Mr. Parag Jain, Mr. R Soorya, Mr. Ritesh Jamunkar, Ms. Prachi Sankhala who extended their kind support and help towards the completion of this project.

It is their help and support, due to which we became able to complete the design and technical report. Without their support this report would not have been possible.

Nilesh Keshwani

B.Tech. IV Year

Department of Computer Science & Engineering

Faculty of Engineering

Medi-Caps University, Indore

Executive Summary

The project is aimed at building a web application named as Distribution Management System. The system is implemented in java platform using spring boot as backend development tool. The main aim of the project is to develop a web application for a telecom company's users for managing the distribution network for the provided inventory. The system aims to automize the process of inventory allocation, user profile management, mapping different users according to their area code (retailers mapped with distributors) and managing inventory.

The intended users of the web application can be classified as: Admin, Distributor and Retailer (following the same hierarchy). Admin can add new inventory, add new distributors, check out their available stock and track the details of retailers and distributors. Distributor can add retailer, request stock from the admin and can allocate stock to retailer. Retailer is the end customer for our application that is mapped to distributor according to area code and can only request stock from his mapped distributor only. This is a brief description of the functionalities of our project.

The technology stack that was used to develop the project is:

- ⦿ Backend: Java EE, Spring Boot and REST services
- ⦿ Frontend: Angular 8
- ⦿ Database: MySQL
- ⦿ Designing: Angular Material, CSS, and Bootstrap
- ⦿ Version Control: Git

Table of Contents

Chapter	Description	Page No.
	Report Approval	ii
	Declaration	iii
	Certificate	iv
	Offer Letter of the Project work-II/Internship	v
	Completion letter/certificate	vi
	Acknowledgement	vii
	Abstract	viii
	Table of Contents	ix
	List of figures	xi
	Abbreviations	xii
Chapter 1	Introduction	
	1.1 Introduction	1
	1.2 Literature Review	2
	1.3 Objectives	3
	1.4 Significance	3
Chapter 2	Project Details	
	2.1 Product Introduction	5
	2.2 Software Requirements	5
	2.3 Functionalities Offered	6
	2.4 System Analysis	6
	2.5 Intended Audience	9
	2.6 Advantages	9
Chapter 3	Project Setup	
	3.1 Angular JS	10
	3.2 Eclipse IDE	10
	3.3 MySQL Workbench	11
	3.4 GitLab	12
	3.5 Visual Studio Code	13
Chapter 4	Technology Description	

	4.1 Angular JS	14
	4.2 Angular Material	19
	4.3 Java Spring Boot	20
	4.4 Microservices	31
	4.5 Mockito Framework	34
	4.6 Junit 5	35
	4.7 MySQL	35
	4.8 Git	36
Chapter 5	System and Designing	
	5.1 Use Case Diagram	37
	5.2 E-R Diagram	38
	5.3 Activity Diagram	39
	5.4 Sequence Diagram	41
	5.5 Component Diagram	42
Chapter 6	Project Screenshots	
	6.1 Login Page	43
	6.2 Admin Profile Page	44
	6.3 Distributor Profile Page	45
	6.4 Retailer Profile Page	46
Chapter 7	Summary and Conclusions	47
Chapter 8	Future Scope	48
	Bibliography	49

List of Figures

Figure Number	Figure Title	Page No.
Figure 4.1.	Angular JS Directives	14
Figure 4.2.	Angular 8 Architecture	16
Figure 4.3.	Spring Boot Framework	20
Figure 4.4.	Spring Boot Architecture	22
Figure 4.5.	Spring Boot Flow Architecture	23
Figure 4.6.	Monolithic vs Microservice Architecture	31
Figure 4.7	Backend to Frontend variations	33
Figure 5.1.	Use Case Diagram	37
Figure 5.2	E-R Diagram	38
Figure 5.3.	Admin Activity Diagram	39
Figure 5.4.	Distributor Activity Diagram	39
Figure 5.5.	Retailer Activity Diagram	40
Figure 5.6.	User Login Sequence Diagram	41
Figure 5.7.	Retailer's Purchase Order Sequence Diagram	42
Figure 5.8.	Component Diagram	42
Figure 6.1.	User's Login Page	43
Figure 6.2.	Admin Profile Page	44
Figure 6.3.	Distributor Profile Page	45
Figure 6.4.	Retailer Profile Page	46

Abbreviations

DMS	Distribution Management System
RAD	Rapid Application Development
CLI	Command Line Interface
REST	Representational State Transfer
RDBMS	Relational Database Management System
IDE	Integrated Development Environment
JPA	Java Persistence API
API	Application Programming Interface

Chapter-1

1.1 Introduction

Management System are now one of the most common types of Web Applications available on the internet, whether it be Students Management System, Inventory Management System, Hospital Management System, Employee Management System, etc. These web applications are either designed as a general-purpose application or for a specific organization. The features and functionalities of these systems depends on the use cases that they are built for, for example a student management system consists of managing the student details, their examination details, their fee details, etc.

With the development of such user centered software the legacy way of handling these services and the problems generated due to human error or frail organization of the institutional data was cut short. These systems could be developed with a common software architecture (Three Tier architecture) but with different services for different needs.

These web applications are developed to deal with common managerial tasks, ease the access of the information to the concerned users, improve the organization, provide data security and integrity, and improve the communication between different users and services.

DMS is a Web-based Application designed as a Management System using Java technology. A DMS is a collection of applications designed to monitor & control the entire distribution network efficiently and reliably. DMS access real-time data and provide all information through a web application in an integrated manner. The main aim of the project is to develop a user interface for inventory maintenance and tracking, stock maintenance, stock transfer, user profile management and purchase management.

The application manages three types of users i.e., Admin, Distributors and Retailers having the hierarchy as mentioned in the order. Each user has some specific services available to them that might be dependent on the user on the hierarchy above him, such as a retailer can't use the application until the distributor has registered him.

1.2 Literature Review

There are many general purpose web applications available on the internet for example the one offered by Apptivo Solutions ([Distribution Management](#) | [Supply Chain Solution](#) | [ERP System - Apptivo](#)) that offers a supply chain based solution with the services like:

- **Manage your orders:** Getting your orders booked instantly by your distributors and customers anywhere, anytime. Organize your workflow with customized features offered by the system. Convert your Sales Orders to Invoices easily without any hassle.
- **Create a Comfortable Delivery Network:** Make your Warehouse System completely automated and with a single click pick products from multiple locations.
- **Secured and Simplified Payments:** Equip Distributors to generate Invoices swiftly at the Point of Sale without any complications.
- **Engaging Inventory Management System:** Apptivo's Inventory Management System makes it easy to gather on-time statistics on the items and their quantities moved in and out of the Inventory. Get to accept or reject a stock transfer request. Keep your inventory updated when an item is added or removed from any warehouse.
- **Detailed Reporting:** Generate timely reports on each action taken in your business.

Arokia IT Pvt. Ltd. offers distribution management software services that accelerates growth, align business goals, and enable peak infrastructural efficiency across business. Their talented teams leverage technological features to provide better controls. They also offer 365x24x7 assistance and facilitate business agility that empowers growth. Some of the key services offered by them are:

- Business automation to substantially minimize the business workflow problems, to automate intricate business process workflows and functionalities.
- Assist to enhance revenues through improved sales efficiency and customer servicing.
- Potent and robust application helps to analyze business distribution and supply chain processes.
- Management of online orders and online sales.

These softwares are not developed to serve a particular type of use case instead a general solution for Distributors based on Supply Chain. The data security cannot be customized according to the customer's needs, integrity of the data cannot be assured either.

These are some general-purpose applications and not to fulfill the needs of our objectives. We considered the common features offered by such software and then designed our own services that fulfilled our objective.

1.3 Objective

The objective of this project is to provide online access to internal and external channel partners of a telecom company by which they can track inventory that will be sold to end customer through different channels. This system provides user interfaces through which user can maintain information regarding receiving inventory, tracking inventory, stock transfer, and allocation of stock (inventory) to Internal or External Channels.

The system should deal with the following modules:

- Inventory Management - Define inventory for Distributors (internal channel) in terms of E –top amounts, recharge coupons, sim cards.
- Channel Profile Management – Retailers (external channel) of a particular area should be map to distributors and profiles of both should be managed in DMS and provided rights according to their role.
- Purchase Management – Retailers should be able to raise Purchase Order in DMS from his Distributor and Distributor should be able to provide inventory to Retailer.

1.4 Significance

The designed web application will be robust and secure with all the objectives covered and common features for user interaction such as Login and Logout along with the several other features such as:

- Inventory Organization: Admin can view the available inventory and add an item to inventory which should be displayed on the buy button to the internal channel partner (distributor).
- View Distributor and holdings: Admin must be able to view the available distributor and his corresponding holdings (retailers mapped according to the area code).
- Track Order History: Admin, distributor and retailer should be able to view the previous transactions (purchase orders) along with date and time of the order.

- Invoice Generation: Users can view the invoice for their purchase orders that will list the details of buyer, seller and the inventory that is bought.
- User Registration: Admin can register a distributor and a distributor can register a retailer (of same area code) with the help of email verification.

Chapter-2

2.1 Product Description

The project is aimed at developing a web-based application named as Distribution Management System that helps in managing the distribution of inventory from the telecom company to various distributors and retailer. DMS helps to view, manage, and record the available inventory, keep a track of the various stakeholders i.e., This project can be categorized in individual aspect as inventory and purchase management system. We are solving the problem of purchase management, user mapping and stock management.

2.2 Software Requirements

The following technologies were used for building the project:

- **Angular JS:** Development platform for building scalable web applications.
- **HTML:** A coding language used to define structure of webpages that a web browser can display.
- **CSS:** It is a design language to improve website appearance and defining a layout.
- **Eclipse for Enterprise:** Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA and others.
- **Java Spring Boot:** Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".
- **MySQL Workbench:** Interactive tool for managing MySQL database, views and schemas.
- **Visual Studio Code:** Code Editor redefined and optimized for building and debugging modern web and cloud applications.
- **Git:** A free open-source distributed version control system designed to handle everything from small to very large project

2.3 Functionalities Offered

➤ **Login and Logout:**

The registered user can login to the application with the right credentials that will be user role, registered email id and password.

➤ **Inventory Management:**

Admin can manage the available inventory after successful login, inventory management includes viewing the inventory and adding new inventory.

➤ **Add Distributor or Retailer:**

Admin should be able to add distributor and a distributor should be able to a retailer of same area code and verify the same through registered user email.

➤ **Manage User Profile:**

User should be able to update the account password.

➤ **Raise Purchase Order:**

Distributor and retailer should be able to raise request for the required inventory on the application.

➤ **View Purchase Order:**

Buyer and sellers should be able to view the purchase order raised for the stock.

➤ **Generate Invoice:**

Buyer must be able to view the invoice corresponding to the stock that he has requested for.

➤ **View Stock:**

User should be able to view its available stock.

2.4 System Analysis

Systems analysis is the process by which an individual studies a system such that an information system can be analyzed, modeled, and a logical alternative can be chosen. Systems analysis projects are initiated for three reasons: problems, opportunities, and directives. The people involved include systems analysts, sponsors, and users. The process by which systems are developed can be described by the systems development life cycle.

Hardware and Software Requirement

Device's Components	Specification
Processor	Intel i3 or Higher
RAM	Minimum 4 GB
Space Required	Less than 300 MB

System Feasibility

Feasibility study includes consideration of all the possible ways to provide a solution to the given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can be easily done based on the future upcoming requirements.

Economic Feasibility

This is a very important aspect to be considered while developing a project. We decided the technology based on minimum possible cost factor-

1. All hardware and software cost has to be borne by the organization.
2. Overall, we have estimated that the benefits the organization is going to receive from the proposed system will surely overcome the initial costs and the later on running cost for system.

Technical Feasibility

This included the study of function, performance and constraints that may affect the ability to achieve an acceptable system. For this feasibility study, we studied complete functionality to be provided in the system, as described in the System Requirement Specification (SRS), and checked if everything was possible using different type of frontend and backend platforms.

Behavioral Feasibility

No doubt the proposed system is a fully real time-based application and we had used GUI also i.e. very user friendly. Besides, proper training has been conducted to let them know the essence of the system so that they feel comfortable with the new system. As far as our study is concerned the clients are comfortable and happy as the system has cut down their loads and doing.

Non-Functional Requirement

Non-functional requirements specify criteria that can be used to judge the operation of a system in particular. These are conditions rather than specific behaviors. While functional requirements define what a system is supposed to do, non-functional requirements define how a system is supposed to be. Non-functional requirements are often called "quality attributes" of a system.

Examples for DMS may include:

- Performance - how much time each page should take to load
- Scalability - will the system be able to handle large volume of users that keeps increasing?
- Capacity - how much storage will be needed?
- Availability - availability and downtime of the application
- Security - this includes security of the content and encryption

2.5 Intended Audience

DMS is a web-based application for a particular Telecom company therefore it only concerns the Channel partners and the admin of the company. The admin can use the application to manage the channel partners and the company offered inventory and the channel partners that are Distributors and Retailers can request to purchase the inventory from the upper hierarchical attendant.

2.6 Advantages

- Distribution monitoring can be done easily.
- Eliminate manual management cost and time.
- Handling of customers is much more convenient.
- Invoice Transfer and generation is convenient.
- Easy to keep track of stock for a particular user.
- Purchase History can be maintained and monitored easily.
- Customers have not to rely on any other 3rd party applications

Chapter-3

3.1 Angular JS

AngularJS is an open-source web application framework. It is used to build Single Page Applications (SPA) projects. It uses HTML for defining the user interfaces while using JavaScript and TypeScript for handling the business logic.

Angular Application Setup:

1. To install Angular CLI you need to have node.js and npm package manager and then you open a command window and run the following command:

npm install -g @angular/cli

2. To create a new workspace and initial starter app: Run the following CLI command:

ng new my-app

where **my-app** is the name of the application.

3. The ng new command prompts you for information about features to include in the initial app. Accept the defaults by pressing the Enter or Return key.
4. Navigate to workspace containing the app.
5. To run the created application, execute the following commands:

cd my-app

ng serve --open

3.2 Eclipse IDE

Eclipse is an integrated development environment (IDE) used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment. We will be using Eclipse IDE for Java EE developers that provides tool for creating Java EE and Web applications including Java IDE, tools for Java EE, JPA, JSF and others.

Setup Eclipse IDE:

1. Download and install jdk (java-development-kit) for your machine (jdk 8 or higher).
2. Download the compatible version for Eclipse IDE (according to the system requirements and the jdk version).
3. Install Eclipse Enterprise Edition by accepting all the licenses and agreement.
4. Since we will be working on Spring Boot, we need to add Spring Suite Tools plugin from Eclipse Marketplace (Taskbar >> Help >> Eclipse Marketplace).
5. After adding the plugin you will be able to see option to build a Spring Starter Project under File menu New option. Click on that provide the package name, project name and project description, select the required dependencies and then click on OK.
6. Ensure a proper Internet Connection and wait till the project is build and dependencies are added to project. Your project setup is complete.

3.3 MySQL Workbench

MySQL Workbench is a unified visual tool for database architects, developers, and DBAs. MySQL Workbench provides data modeling, SQL development, and comprehensive administration tools for server configuration, user administration, backup, and much more. MySQL Workbench is available on Windows, Linux, and Mac OS.

Setting up MySQL Workbench:

1. Open MySQL website and in the download, options select MySQL installer for windows.
2. Choose the desired installer and click on download and then run the setup wizard.
3. Then allow the permissions and select custom in setup type windowpane. Then select MySQL server, MySQL Workbench and MySQL shell.
4. Then set up MySQL root password and you are done with your installation.

3.4 Git Lab

GitLab's application offers functionality to collaboratively plan, build, secure, and deploy software as a complete DevOps Platform. GitLab is highly scalable and can be hosted on-premises or on cloud storage. It also includes a wiki, issue-tracking, IDE, and CI/CD pipeline features. GitLab, like GitHub, also offers a free GitLab Pages product for hosting static webpages.

To create a blank project:

1. On the top bar, select Menu > Projects > Create new project. Select Create blank project.
2. Enter the project details:
 - a. In the Project name field, enter the name of your project. You cannot use special characters at the start or end of a project name.
 - b. In the Project slug field, enter the path to your project. The GitLab instance uses the slug as the URL path to the project. To change the slug, first enter the project name, then change the slug.
 - c. In the Project description (optional) field, enter the description of your project's dashboard.
 - d. In the Project target (optional) field, select your project's deployment target. This information helps GitLab better understand its users and their deployment requirements.
 - e. To modify the project's viewing and access rights for users, change the Visibility Level.
 - f. To create README file so that the Git repository is initialized, has a default branch, and can be cloned, select Initialize repository with a README.
 - g. To analyze the source code in the project for known security vulnerabilities, select Enable Static Application Security Testing (SAST).
3. Select Create project.

3.5 Visual Studio Code

Visual Studio Code commonly known as VS Code, is a source-code editor made by Microsoft for Windows, Linux and MacOS. Its features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring and embedded Git. Users can change the theme, keyboard shortcuts, preferences and install plugins and extensions that provide additional functionality.

Instead of a project system, it allows users to open one or more directories, which can then be saved in workspaces for future reuse. This allows it to operate as a language-agnostic code editor for any language. It supports many programming languages and a set of features that differs per language. Unwanted files and folders can be excluded from the working tree via settings.

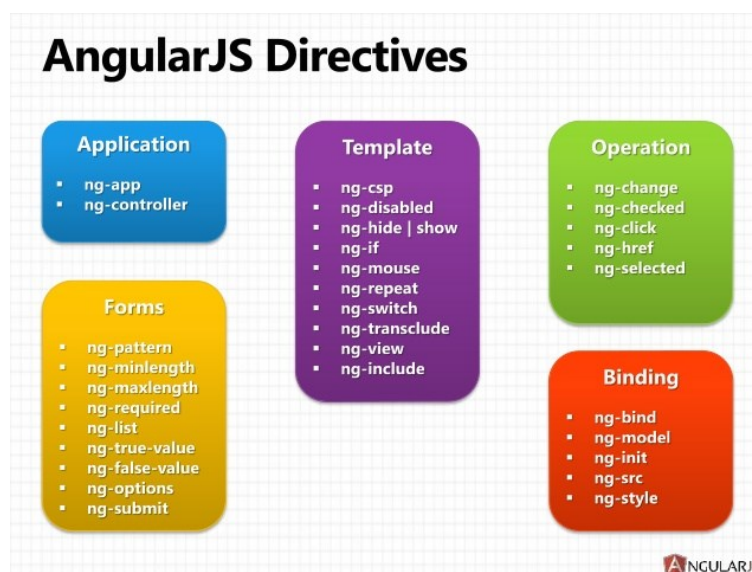
Chapter-4

4.1 Angular JS

AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. AngularJS's data binding and dependency injection eliminate much of the code you would otherwise have to write. And it all happens within the browser, making it an ideal partner with any server technology.

Features of AngularJs:

1. **Data Binding** : In an angular application, we don't need to write separate code to perform the data binding (synchronization of data between business logic and view of the application) functionality. By adding some snippets of code, we can easily bind data from HTML control to application data. Any extra code is not written to bind with HTML control.
2. **Architecture** : An angular application is built using MVC architecture that stands for Model View and Controller. It separates the application into three parts model part, view part and controller part as per the components of MVC architecture. Using this,



architecture presentation part, logic part and application data part is split into the separate section which allows managing of application in a very fluent manner.

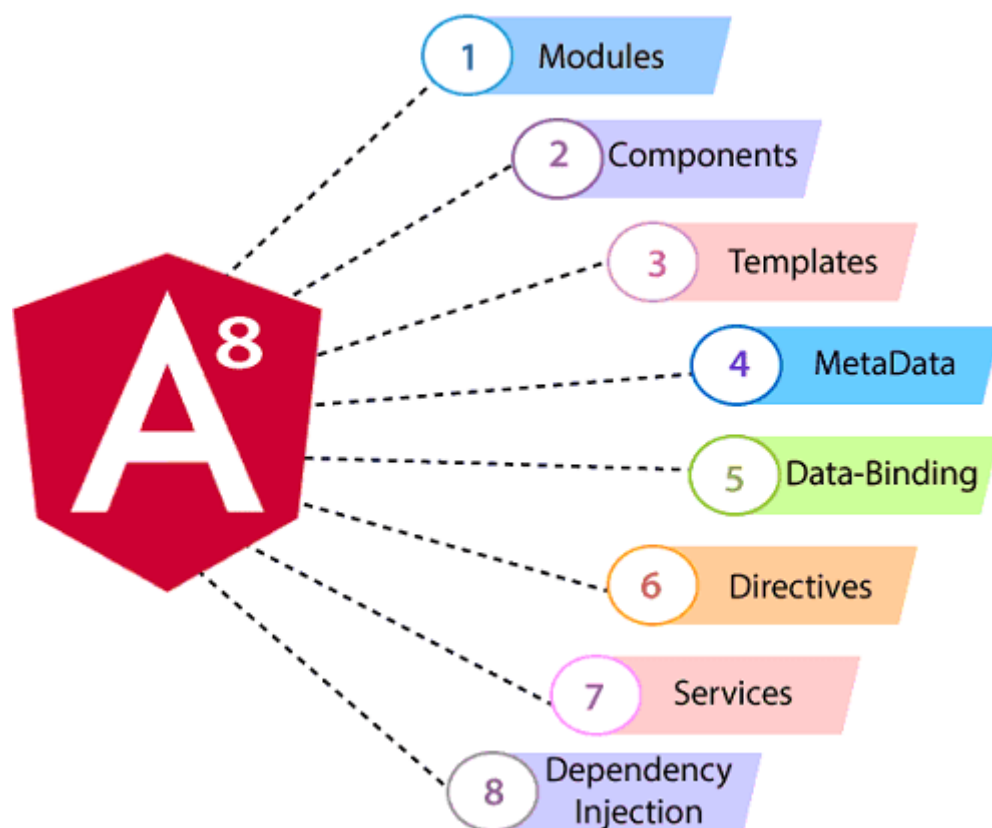
3. **Directives** : View of AngularJS, mix data from a model into HTML templates. Angular JS directives are used for the same purpose. It tells how to combine data into the HTML template. With the use of directive, we can provide extra functionality to our angular application. Angular provides a way to create custom directives too. There is a list of angular JS directives:
4. **Not Browser Specific** : Angular applications are not browser specific means there is no browser constraint on an angular application.
5. **Codeless** : A programmer can write less and perform more functionalities with the same code. Filters in angular provide the functionality of write less do more. The various filters in angular are uppercase, lowercase, currency etc. You can use the filter and easily format the data.
6. **Speed and Performance** : Speed and performance of angular are faster because of three things:
 - **Code Generation** – When you are writing code using angular, it converts your template into a highly optimized code that gives you an advantage of handwritten code with the productivity of framework.
 - **Universal** – The first view of your application on .net, PHP, node.js and other servers that is till now dependent on HTML CSS for their front end serve using angular.
 - **Code Splitting** – Its new component router loads angular app quickly. It provides the ability of automatic code splitting too. Therefore, only that code is loaded which is requested to render the view.
7. **Dependency Injection** : This built-in injection helps in developing the application easily as well as it is easy to understand. It helps an application easier to test. Whenever angular JS detect that you need a service then it immediately provides an instance for that. It allows you to ask for your dependencies rather than having to go look for them or making it by yourself.
8. **Deep Linking** : It allows to bookmark the web page. The page gets saved by its URL without getting its state changed. Whenever the request is made by a user for that page it will get displayed in the same state as before.

9. **Routing :** Routing allows the switching between views. Being a single page application ngRoute directive provided by angular, helps a user to navigate from one view to another, but the application will remain single page. It means without reloading the angular application you can switch to different pages in your application.

10. **Productivity :**

- **Template** – Template in the angular application allows a developer to create user interface quickly as it provides simple and powerful template syntax.
- **Angular CLI** – It is a command line tool. It starts building an application very fast. It adds components, tests it and then deploys it instantly.
- **IDE** – Intelligent code completion is possible through IDE. It will find instant errors and provides other feedbacks too.

Angular 8 Architecture



Angular 8 is written in TypeScript. It implements core and optional functionality as a set of TypeScript libraries that you can import into your apps.

The basic building blocks of an Angular application are NgModules, which provide a compilation context for components. NgModules collect related code into functional sets; an Angular app is defined by a set of NgModules. An app always has at least a root module that enables bootstrapping, and typically has many more feature modules.

- Components define views, which are sets of screen elements that Angular can choose among and modify according to your program logic and data.
- Components use services, which provide specific functionality not directly related to views. Service providers can be injected into components as dependencies, making your code modular, reusable, and efficient.

Angular 8 Components

Components and services both are simply classes with decorators that mark their types and provide metadata which guide Angular to do things. Every Angular application always has at least one component known as root component that connects a page hierarchy with page DOM. Each component defines a class that contains application data and logic, and is associated with an HTML template that defines a view to be displayed in a target environment.

Modules

Angular 8 NgModules are different from other JavaScript modules. Every Angular 8 app has a root module known as AppModule. It provides the bootstrap mechanism that launches the application.

- Angular 8 NgModules import the functionalities from other NgModules just like other JavaScript modules.
- NgModules allow their own functionality to be exported and used by other NgModules. For example, if you want to use the router service in your app, you can import the Router NgModule.

Routing

Router is an NgModule, which provides a service that facilitates developers to define a navigation path among the different application states and view hierarchies in their app. It works in the same way as a browser's navigation works. i.e.:

- Enter a URL in the address bar and the browser will navigate to that corresponding page.
- Click the link on a page and the browser will navigate to a new page.
- Click the browser's back or forward buttons and the browser will navigate backward or forward according to your seen history pages.

Services and Dependency Injection

Dependencies are services or objects that a class needs to perform its function. Dependency injection, or DI, is a design pattern in which a class requests dependencies from external sources rather than creating them. Angular's DI framework provides dependencies to a class upon instantiation. Use Angular DI to increase flexibility and modularity in your applications.

Dependency Injection (DI) is used to make your component classes lean and efficient. DI does not fetch data from the server, validate user input, or log directly to the console; it simply renders such tasks to services. In Angular 8, developers create a service class for data or logic that is not associated with a specific view, and they want to share across components.

Angular 8 Directives

The Angular 8 directives are used to manipulate the DOM. By using Angular directives, you can change the appearance, behaviour or a layout of a DOM element. It also helps you to extend HTML. Angular 8 directives can be classified in 3 categories based on how they behave:

- **Component Directives:** Component directives are used in main class. They contain the detail of how the component should be processed, instantiated and used at runtime.
- **Structural Directives:** Structural directives start with a * sign. These directives are used to manipulate and change the structure of the DOM elements. For example, *ngIf directive, *ngSwitch directive, and *ngFor directive.
- **Attribute Directives:** Attribute directives are used to change the look and behaviour of the DOM elements. For example: ngClass directive, and ngStyle directive etc.

Data Binding in Angular 8

Data binding is the core concept of Angular 8 and used to define the communication between a component and the DOM. It is a technique to link your data to your view layer. In simple words, you can say that data binding is a communication between your typescript code of your component and your template which user sees. It makes easy to define interactive applications without worrying about pushing and pulling data.

Data binding can be either one-way data binding or two-way data binding. Angular provides four types of data binding and they are different on the way of data flowing.

- String Interpolation
- Property Binding
- Event Binding
- Two-way binding

4.2 Angular Material

Angular Material is a UI library component developed by Google in 2014. It is specially designed for AngularJS developers. It helps to design the application in a structured manner. Its components help to construct attractive, consistent, and functional web pages and web applications. It is used to create a responsive and faster website.

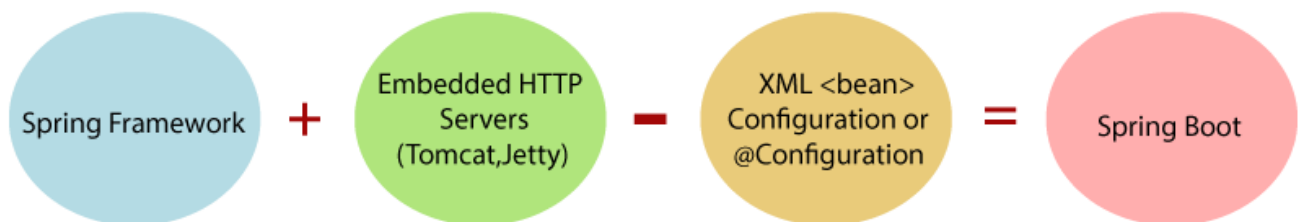
It attracts users and access the elements or components present in our application. It also helps to design our applications with unique styles and shapes. These components make the application or website more consistent and design responsive. It combines the classic principles of successful design with innovation and technology.

Features of Angular Material

- It is an In-built responsive design.
- Angular Material has standard CSS.
- The new version of UI Components which are **buttons, checkboxes, and text fields** is used to follow Angular Material Design concepts.
- It has specialized features such as **cards, toolbar, speed dial, side nav, swipe, side nav**, and more.
- It is a cross-platform browser and it is used to create web components.

4.3 Java Spring Boot

Spring Boot is an open source, microservice-based Java web framework. The Spring Boot framework creates a fully production-ready environment that is completely configurable using



its prebuilt code within its codebase. The microservice architecture provides developers with a fully enclosed application, including embedded application servers. It provides an easier and faster way to set up, configure, and run both simple and web-based applications. It is a Spring module that provides the RAD (Rapid Application Development) feature to the Spring Framework. It is used to create a stand-alone Spring-based application that you can just run because it needs minimal Spring configuration.

Along with the Spring Boot Framework, many other Spring sister projects help to build applications addressing modern business needs. There are the following Spring sister projects are as follows:

- **Spring Data:** It simplifies data access from the relational and NoSQL databases.
- **Spring Batch:** It provides powerful batch processing.
- **Spring Security:** A security framework provides robust security to applications.
- **Spring Social:** It supports integration with social networking like LinkedIn.
- **Spring Integration:** It is an implementation of Enterprise Integration Patterns. It facilitates integration with other enterprise applications using lightweight messaging and declarative adapters.

Advantages of Spring Boot

- The primary advantage is spring boot offers an effortless way to create spring-based applications using JAVA or Groovy.
- The greatest advantage is the reduced time. Spring Boot minimizes the time spent in developing and increasing productivity.
- Also, it helps in reducing all the manual work of writing annotations, boilerplate codes and XML configurations.
- It has made the integration of Spring Boot Application with its Spring Ecosystem, which includes Spring Security, Spring Data, Spring JDBC and Spring ORM easy.
- By following the ‘Opinionated Defaults Configuration’, it also reduces the efforts of developers.
- Developers have easy access to Embedded HTTP servers such as Jetty, Tomcat and also easily test the web applications effortlessly.
- Alongside, it also provides a lot of plugins which aids in effortless development and testing of Spring Boot application build with the help of tools like Gradle and Maven.
- Lastly, it provides a plugin that has made working with embedded and in-memory databases very smoothly and readily.

Prerequisite of Spring Boot

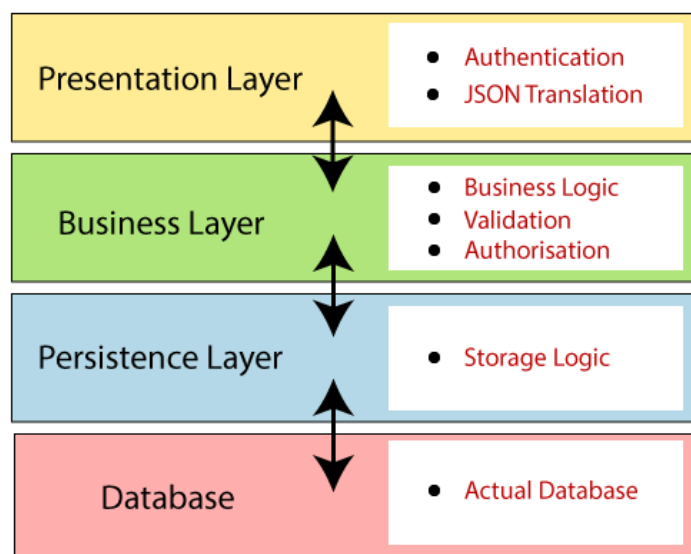
To create a Spring Boot application, following are the prerequisites. In this tutorial, we will use Spring Tool Suite (STS) IDE.

- Java 1.8
- Maven 3.0+
- Spring Framework 5.0.0.BUILD-SNAPSHOT
- An IDE (Spring Tool Suite) is recommended.

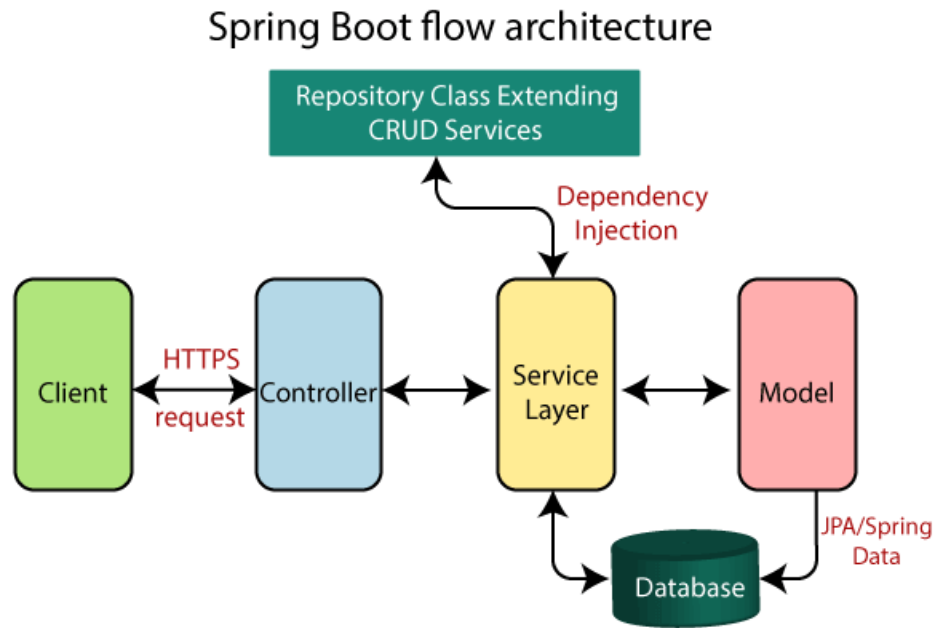
Spring Boot Architecture

There are **four** layers in Spring Boot are as follows:

- Presentation Layer
- Business Layer
- Persistence Layer
- Database Layer



Spring Boot Flow Architecture



Spring Boot Annotations

Spring Boot Annotations is a form of metadata that provides data about a program. In other words, annotations are used to provide **supplemental** information about a program. It is not a part of the application that we develop. It does not have a direct effect on the operation of the code they annotate. It does not change the action of the compiled program.

- **@Required:** It applies to the bean setter method. It indicates that the annotated bean must be populated at configuration time with the required property, else it throws an exception `BeanInitializationException`.
- **@Autowired:** Spring provides annotation-based auto-wiring by providing `@Autowired` annotation. It is used to autowire spring bean on setter methods, instance variable, and constructor.
- **@Configuration:** It is a class-level annotation. The class annotated with `@Configuration` used by Spring Containers as a source of bean definitions.
- **@ComponentScan:** It is used when we want to scan a package for beans. It is used with the annotation `@Configuration`.

Spring Framework Stereotype Annotations

- **@Component:** It is a class-level annotation. It is used to mark a Java class as a bean. A Java class annotated with @Component is found during the classpath. The Spring Framework pick it up and configure it in the application context as a Spring Bean.
- **@Controller:** The @Controller is a class-level annotation. It is a specialization of @Component. It marks a class as a web request handler. It is often used to serve web pages. By default, it returns a string that indicates which route to redirect. It is mostly used with @RequestMapping annotation.
- **@Service:** It is also used at class level. It tells the Spring that class contains the **business logic**.
- **@Repository:** It is a class-level annotation. The repository is a DAOs (Data Access Object) that access the database directly. The repository does all the operations related to the database.

Spring MVC and REST Annotations

- **@RequestMapping:** It is used to map the web requests. It has many optional elements like consumes, header, method, name, params, path, produces, and value. We use it with the class as well as the method.
- **@GetMapping:** It maps the HTTP GET requests on the specific handler method. It is used to create a web service endpoint that fetches It is used instead of using: @RequestMapping(method = RequestMethod.GET)
- **@PostMapping:** It maps the HTTP POST requests on the specific handler method. It is used to create a web service endpoint that creates It is used instead of using: @RequestMapping(method = RequestMethod.POST)
- **@PutMapping:** It maps the HTTP PUT requests on the specific handler method. It is used to create a web service endpoint that creates or updates It is used instead of using: @RequestMapping(method = RequestMethod.PUT)

- **@DeleteMapping:** It maps the HTTP DELETE requests on the specific handler method. It is used to create a web service endpoint that deletes a resource. It is used instead of using: `@RequestMapping(method = RequestMethod.DELETE)`
- **@RequestBody:** It is used to bind HTTP request with an object in a method parameter. Internally it uses HTTP MessageConverters to convert the body of the request. When we annotate a method parameter with `@RequestBody`, the Spring framework binds the incoming HTTP request body to that parameter.
- **@ResponseBody:** It binds the method return value to the response body. It tells the Spring Boot Framework to serialize a return an object into JSON and XML format.
- **@PathVariable:** It is used to extract the values from the URI. It is most suitable for the RESTful web service, where the URL contains a path variable. We can define multiple `@PathVariable` in a method.
- **@RequestParam:** It is used to extract the query parameters from the URL. It is also known as a query parameter. It is most suitable for web applications. It can specify default values if the query parameter is not present in the URL.
- **@RestController:** It can be considered as a combination of `@Controller` and `@ResponseBody` annotations. The `@RestController` annotation is itself annotated with the `@ResponseBody` annotation. It eliminates the need for annotating each method with `@ResponseBody`.
- **@RequestAttribute:** It binds a method parameter to request attribute. It provides convenient access to the request attributes from a controller method. With the help of `@RequestAttribute` annotation, we can access objects that are populated on the server-side.

Spring Boot Dependency Management

Spring Boot manages dependencies and configuration automatically. Each release of Spring Boot provides a list of dependencies that it supports. The list of dependencies is available as a part of the Bills of Materials (`spring-boot-dependencies`) that can be used with Maven

Advantages of Dependency Management

- It provides the centralization of dependency information by specifying the Spring Boot version in one place. It helps when we switch from one version to another.
- It avoids mismatch of different versions of Spring Boot libraries.
- We only need to write a library name with specifying the version. It is helpful in multi-module projects.

Spring Boot Application Properties

Spring Boot provides various properties that can be configured in the `application.properties` file. The properties have default values. We can set a property for the Spring Boot application. Spring Boot also allows us to define our own property if required.

The `application.properties` file allows us to run an application in a different environment. In short, we can use the `application.properties` file to:

- Configure the Spring Boot framework
- define our application custom configuration properties

Spring Boot Starters

Spring Boot provides a number of starters that allow us to add jars in the classpath. Spring Boot built-in starters make development easier and rapid. Spring Boot Starters are the dependency descriptors.

In the Spring Boot Framework, all the starters follow a similar naming pattern: `spring-boot-starter-*`, where `*` denotes a particular type of application. For example, if we want to use Spring and JPA for database access, we need to include the `spring-boot-starter-data-jpa` dependency in our `pom.xml` file of the project.

Spring Boot Starter Web

Starter of Spring web uses Spring MVC, REST and Tomcat as a default embedded server. The single `spring-boot-starter-web` dependency transitively pulls in all dependencies related to web development. It also reduces the build dependency count.

The `spring-boot-starter-web` auto-configures the following things that are required for the web development:

- Dispatcher Servlet
- Error Page
- Web JARs for managing the static dependencies
- Embedded servlet container

Spring Data JPA

Spring Data JPA adds a layer on the top of JPA. It means, Spring Data JPA uses all features defined by JPA specification, especially the entity, association mappings, and JPA's query capabilities. Spring Data JPA adds its own features such as the no-code implementation of the repository pattern and the creation of database queries from the method name.

Spring Data is a high-level Spring Source project. Its purpose is to unify and easy access to the different kinds of persistence stores, both relational database systems, and NoSQL data stores.

Spring Data JPA Features

There are **three** main features of Spring Data JPA are as follows:

- **No-code repository:** It is the most popular persistence-related pattern. It enables us to implement our business code on a higher abstraction level.

- **Reduced boilerplate code:** It provides the default implementation for each method by its repository interfaces. It means that there is no longer need to implement read and write operations.
- **Generated Queries:** Another feature of Spring Data JPA is the **generation of database queries** based on the method name. If the query is not too complex, we need to define a method on our repository interface with the name that starts with **findBy**.

Spring Data Repository

Spring Data JPA provides **three** repositories are as follows:

- **CrudRepository:** It offers standard create, read, update, and delete. It contains method like `findOne()`, `findAll()`, `save()`, `delete()`, etc.
- **PagingAndSortingRepository:** It extends the `CrudRepository` and adds the `findAll` methods. It allows us to sort and retrieve the data in a paginated way.
- **JpaRepository:** It is a JPA specific repository. It is defined in `Spring Data Jpa`. It extends the both repository `CrudRepository` and `PagingAndSortingRepository`. It adds the JPA-specific methods, like `flush()` to trigger a flush on the persistence context.

Spring Boot CRUD Operations

The **CRUD** stands for **Create, Read/Retrieve, Update, and Delete**. These are the four basic functions of the persistence storage. The CRUD operation can be defined as user interface conventions that allow view, search, and modify information through computer-based forms and reports. CRUD is data-oriented and the standardized use of **HTTP action verbs**. HTTP has a few important verbs.

- **POST:** Creates a new resource
- **GET:** Reads a resource
- **PUT:** Updates an existing resource
- **DELETE:** Deletes a resource

Spring Boot CrudRepository

Spring Boot provides an interface called `CrudRepository` that contains methods for CRUD operations. It is defined in the package `org.springframework.data.repository`. It extends the `Spring Data Repository` interface. It provides generic Crud operation on a repository.

Spring Boot JpaRepository

`JpaRepository` provides JPA related methods such as flushing, persistence context, and deletes a record in a batch. . It is defined in the package “`org.springframework.data.jpa.repository`” `JpaRepository` extends both `CrudRepository` and `PagingAndSortingRepository`.

Spring Security

Spring Security is a framework which provides various security features like: authentication, authorization to create secure Java Enterprise Applications. It overcomes all the problems that come during creating non spring security applications and manage new server environment for the application. This framework targets two major areas of application are authentication and authorization. Authentication is the process of knowing and identifying the user that wants to access.

Spring Security framework supports wide range of authentication models. These models either provided by third parties or framework itself. Spring Security supports integration with all of these technologies.

- HTTP BASIC authentication headers
- HTTP Digest authentication headers
- HTTP X.509 client certificate exchange
- LDAP (Lightweight Directory Access Protocol)
- Form-based authentication
- OpenID authentication

- Automatic remember-me authentication
- Kerberos
- JOSSO (Java Open Source Single Sign-On)
- AppFuse
- AndroMDA
- Mule ESB
- DWR(Direct Web Request)

The beauty of this framework is its flexible authentication nature to integrate with any software solution. Sometimes, developers want to integrate it with a legacy system that does not follow any security standard, there Spring Security works nicely.

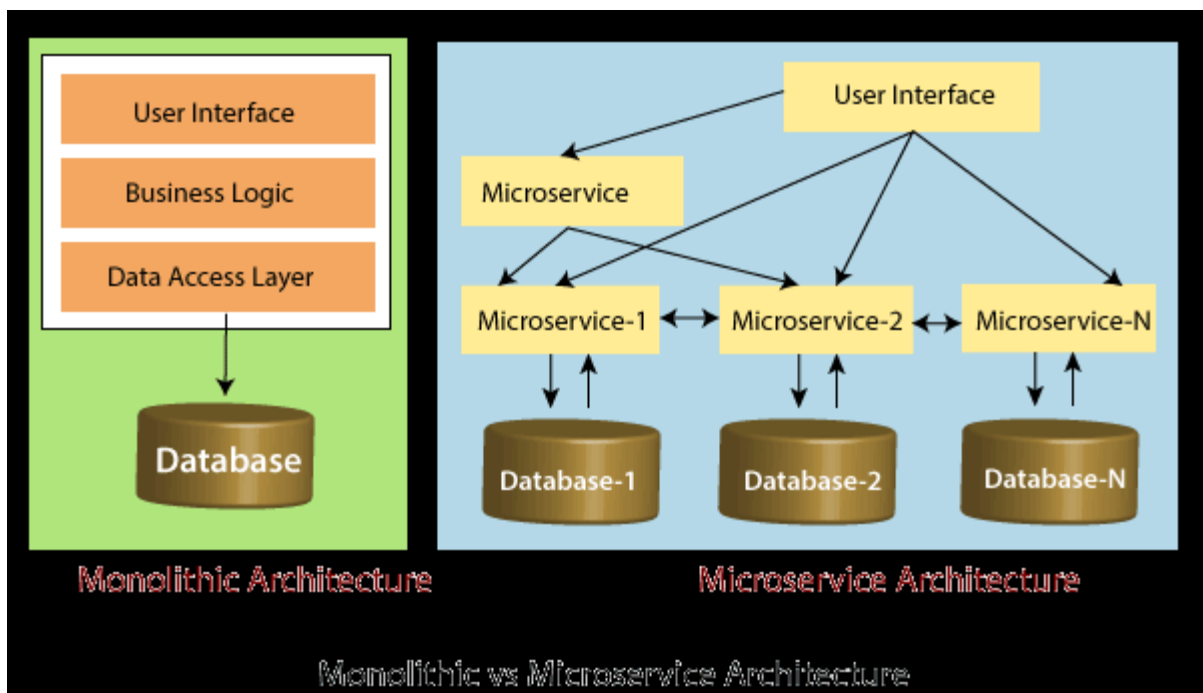
Advantages

Spring Security has numerous advantages. Some of that are given below.

- Comprehensive support for authentication and authorization.
- Protection against common tasks
- Servlet API integration
- Integration with Spring MVC
- Portability
- CSRF protection
- Java Configuration support

4.4 Microservices

The microservice defines an approach to the architecture that divides an application into a pool of loosely coupled services that implements business requirements. It is next to Service-Oriented Architecture (SOA). The most important feature of the microservice-based architecture is that it can perform continuous delivery of a large and complex application. Microservice helps in breaking the application and build logically independent smaller applications.



If we change in one microservice, it does not affect the other services. These services communicate with each other by using lightweight protocols such as HTTP or REST or messaging protocols.

Principles of Microservices

- Single Responsibility principle
- Modelled around business domain
- Isolate Failure
- Infrastructure automation
- Deploy independently

Advantages of Microservices

- Microservices are self-contained, independent deployment module.
- The cost of scaling is comparatively less than the monolithic architecture.
- Microservices are independently manageable services. It can enable more and more services as the need arises. It minimizes the impact on existing service.
- It is possible to change or upgrade each service individually rather than upgrading in the entire application.
- Microservices allows us to develop an application which is organic (an application which latterly upgrades by adding more functions or modules) in nature.
- It enables event streaming technology to enable easy integration in comparison to heavyweight interposes communication.
- Microservices follows the single responsibility principle.

Disadvantages of Microservices

- Microservices has all the associated complexities of the distributed system.
- There is a higher chance of failure during communication between different services.
- Difficult to manage a large number of services.
- The developer needs to solve the problem, such as network latency and load balancing.
- Complex testing over a distributed environment.

Spring Boot - Eureka Server

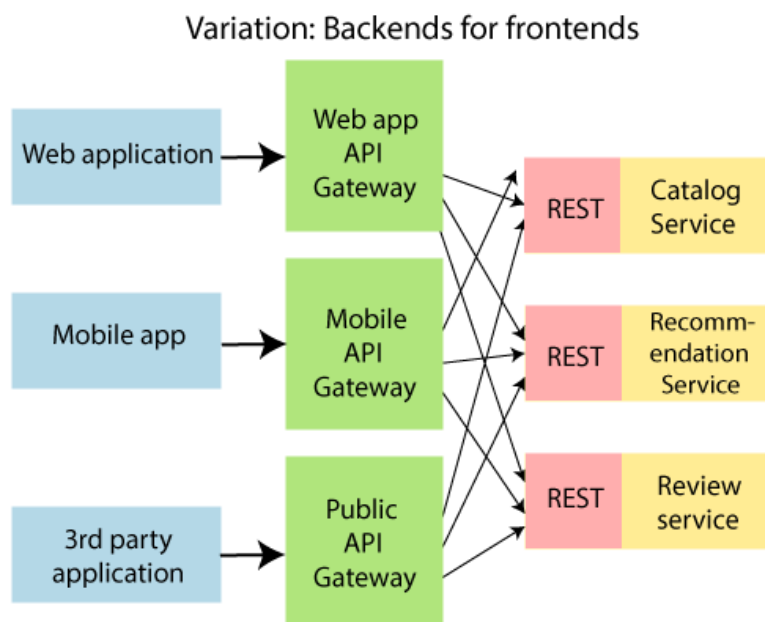
Eureka Server is an application that holds the information about all client-service applications. Every Micro service will register into the Eureka server and Eureka server knows all the client applications running on each port and IP address. Eureka Server is also known as Discovery Server.

Microservices Service Registry

A service registry is a database used to keep track of the available instances of each microservice in an application. The service registry needs to be updated each time a new

service comes online and whenever a service is taken offline or becomes unavailable. This can be achieved with either self-registration or third-party registration.

API Gateway for Microservices



API gateway is the essential component to work with when it comes to microservices application patterns. API gateway represents a single entry point into the system, providing the request routing, security, protocol translation, and composition of microservices in an application. API gateway also enables to mark the failure in the backend services of the application by returning the default data.

API Gateway is also responsible for request routing, composition, and protocol translation. All the requests made by the client go through the API Gateway. After that, the API Gateway routes requests to the appropriate microservice.

Making use of microservices API gateway enables efficiency and speed of the data transfer. Using the declarative style to write the application makes it possible for the API gateway to fulfil the clients' requests and share the desired responses.

An API Gateway includes:

- Security
- Caching
- API composition and processing
- Managing access quotas
- API health monitoring
- Versioning
- Routing

4.5 Mockito Framework

Mockito is a mocking framework. It is a Java-based library used to create simple and basic test APIs for performing unit testing of Java applications. It can also be used with other frameworks such as JUnit and TestNG.

Mocking is a process of developing the objects that act as the mock or clone of the real objects. In other words, mocking is a testing technique where mock objects are used instead of real objects for testing purposes. Mockito is a Java-based mocking framework used for unit testing of Java application. Mockito plays a crucial role in developing testable applications.

Mockito mock() method

It is used to create mock objects of a given class or interface. Mockito contains five mock() methods with different arguments. When we didn't assign anything to mocks, they will return default values.

Mockmvc

MockMVC class is part of Spring MVC test framework which helps in testing the controllers explicitly starting a Servlet container. In this MockMVC tutorial, we will use it along with Spring boot's WebMvcTest class to execute Junit testcases which tests REST controller methods written for Spring boot 2 hateoas example. tests using MockMvc are in the

boundaries between unit and integration tests. They aren't unit tests because endpoints are tested in integration with a Mocked MVC container with mocked inputs and dependencies

4.6 JUnit 5

JUnit is a Java unit testing framework that's one of the best test methods for regression testing. An open-source framework, it is used to write and run repeatable automated tests. A JUnit test is a method contained in a class which is only used for testing. This is called a Test class. To define that a certain method is a test method, annotate it with the `@Test` annotation. This method executes the code under test.

Features of JUnit

- JUnit is an open source framework, which is used for writing and running tests.
- Provides annotations to identify test methods.
- Provides assertions for testing expected results.
- Provides test runners for running tests.

JUnit 5 is the next generation of JUnit. The goal is to create an up-to-date foundation for developer-side testing on the JVM. This includes focusing on Java 8 and above, as well as enabling many different styles of testing. JUnit 5 leverages features from Java 8 or later, like lambda functions, making tests more powerful and easier to maintain. JUnit 5 has added some very useful new features for describing, organizing, and executing tests. For instance, tests get better display names and can be organized hierarchically

4.7 MySQL

MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by Oracle Company. MySQL is a Relational Database Management System (RDBMS) software that provides many things, which are as follows:

- It allows us to implement database operations on tables, rows, columns, and indexes.
- It defines the database relationship in the form of tables (collection of rows and columns), also known as relations.
- It provides the Referential Integrity between rows or columns of various tables.
- It allows us to update the table indexes automatically.
- It uses many SQL queries and combines useful information from multiple tables for the end-users.

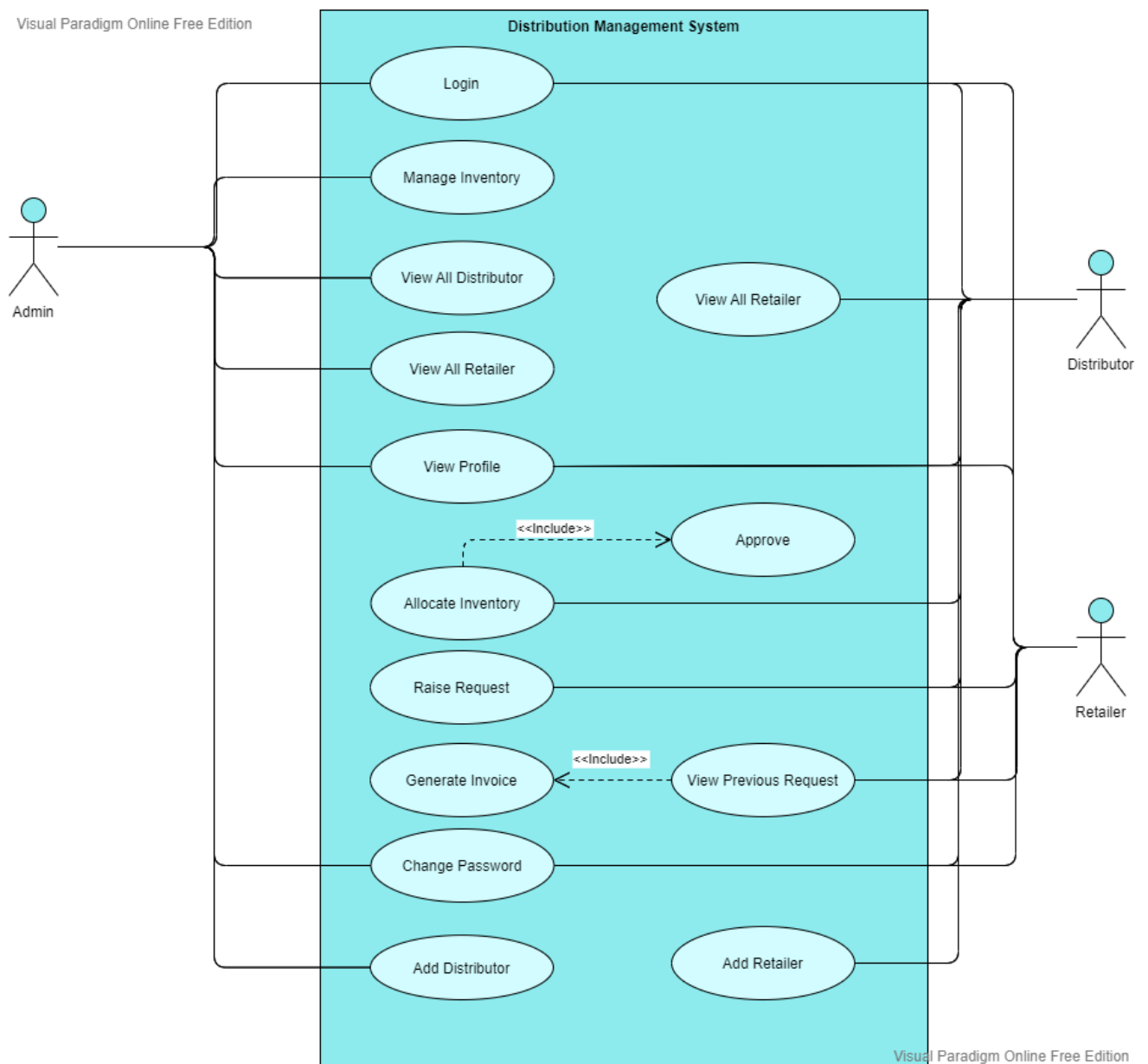
4.8 Git

Git is software for tracking changes in any set of files, folders, directories that is been coordinated among various set of peoples. Git helps in coordinating between all such stakeholders so they can work among themselves and collaboratively develop source code during software development. It aims at increasing speed, maintaining data integrity and provide support for distributed and non-linear workflows.

Chapter-5

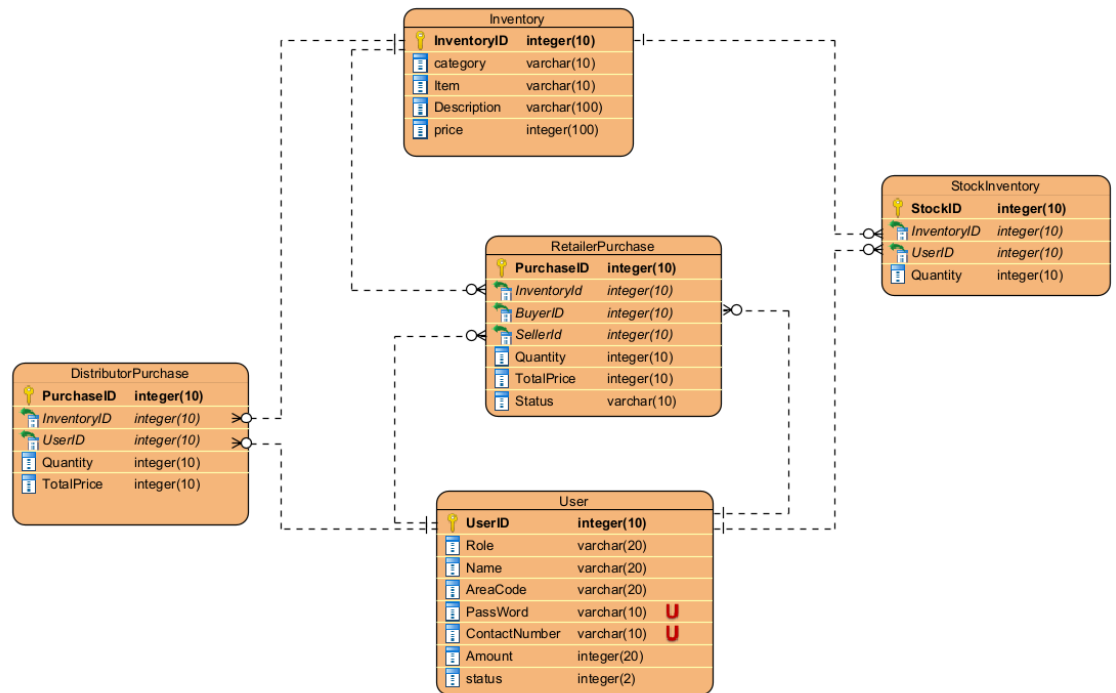
5.1 Use Case Diagram

Use Case diagram represent the interactions between various actors of our application that are admin, distributor, and retailer. It also depicts the offered functionalities to the actors. Use case diagram for DMS is:



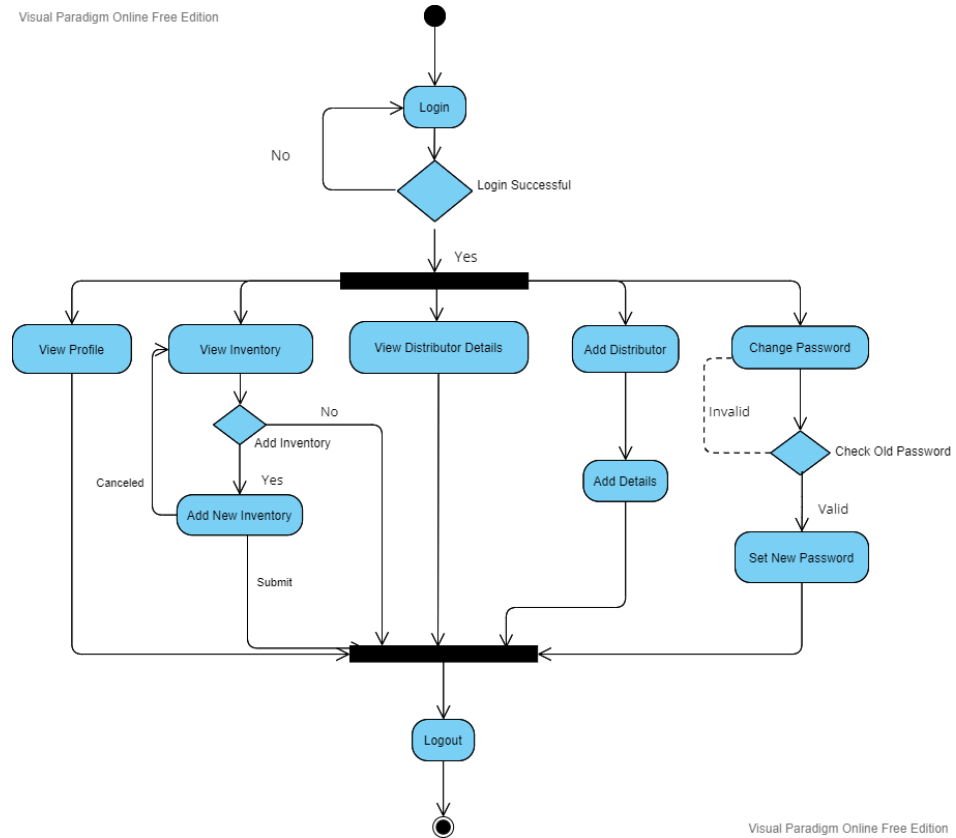
5.2 E-R Diagram

It represents the database view of DMS, the tables created to store the data, the name of table attributes along with its datatype and constraints. The following is the ER diagram for our project:

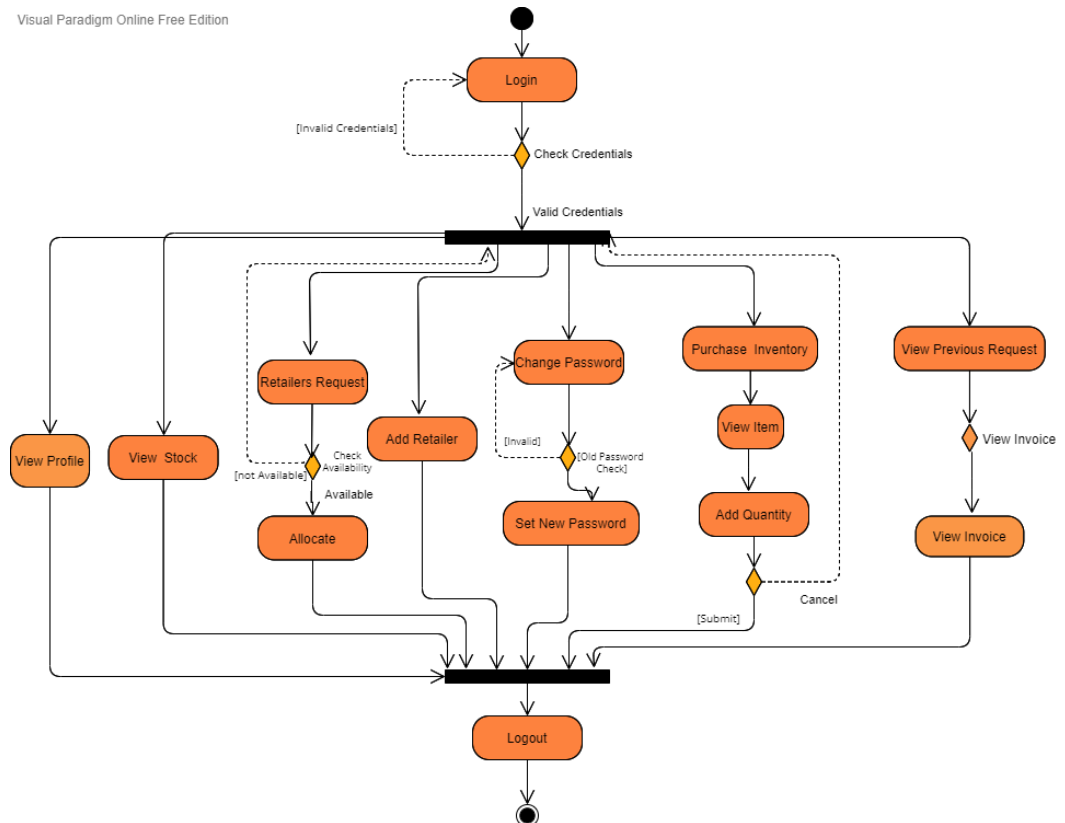


5.3 Activity Diagrams

Visual Paradigm Online Free Edition

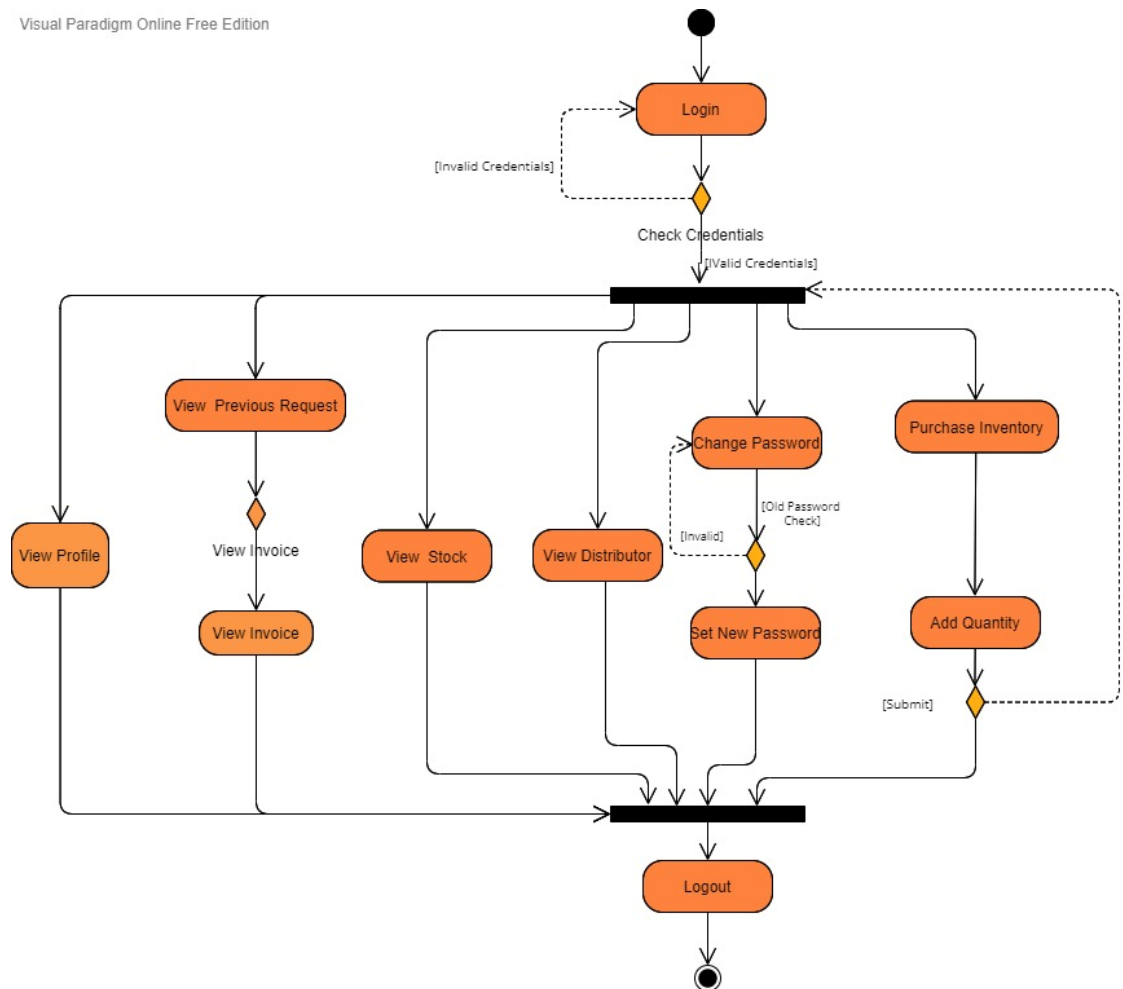


Visual Paradigm Online Free Edition



Distributor Activity Diagram

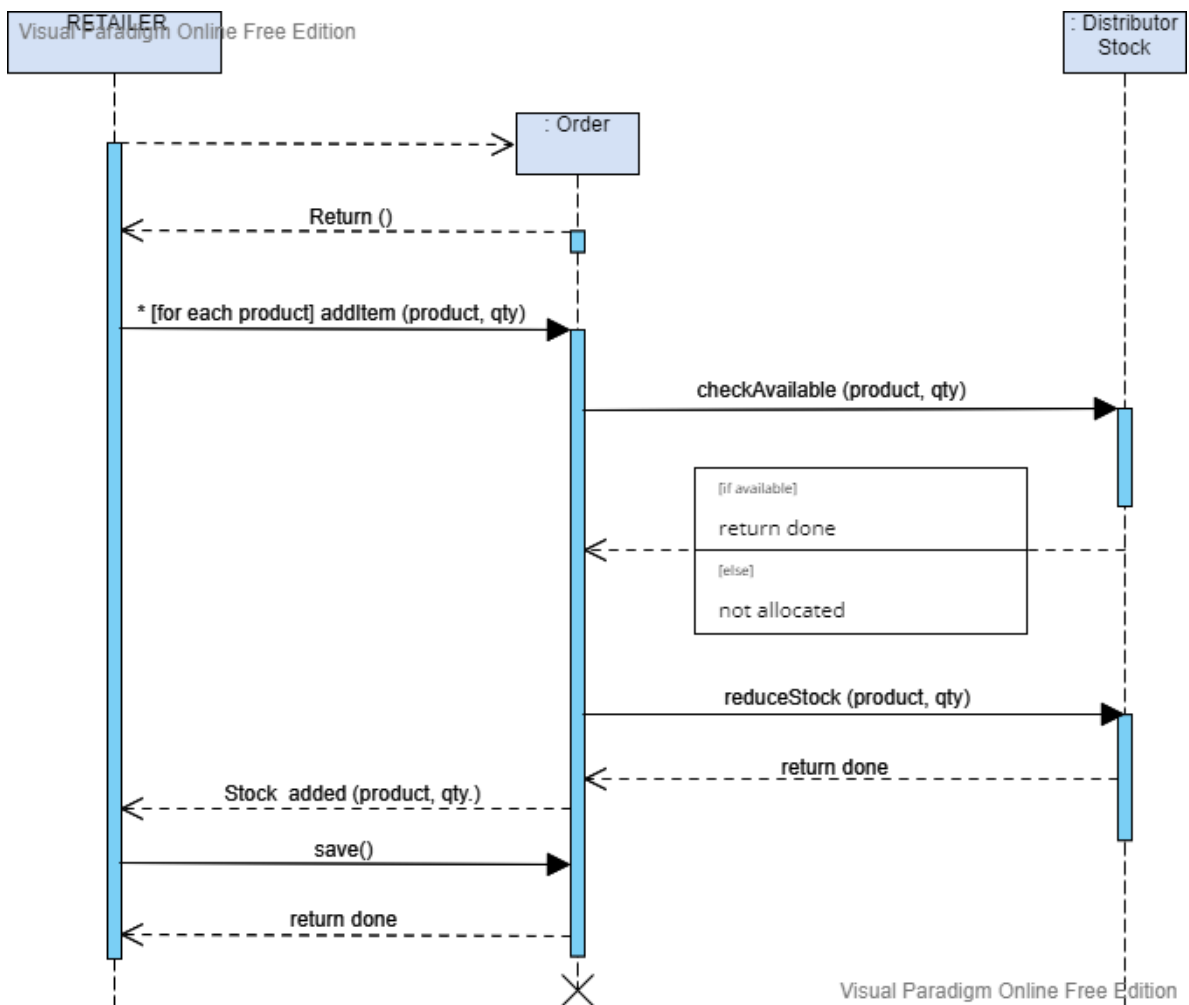
Visual Paradigm Online Free Edition

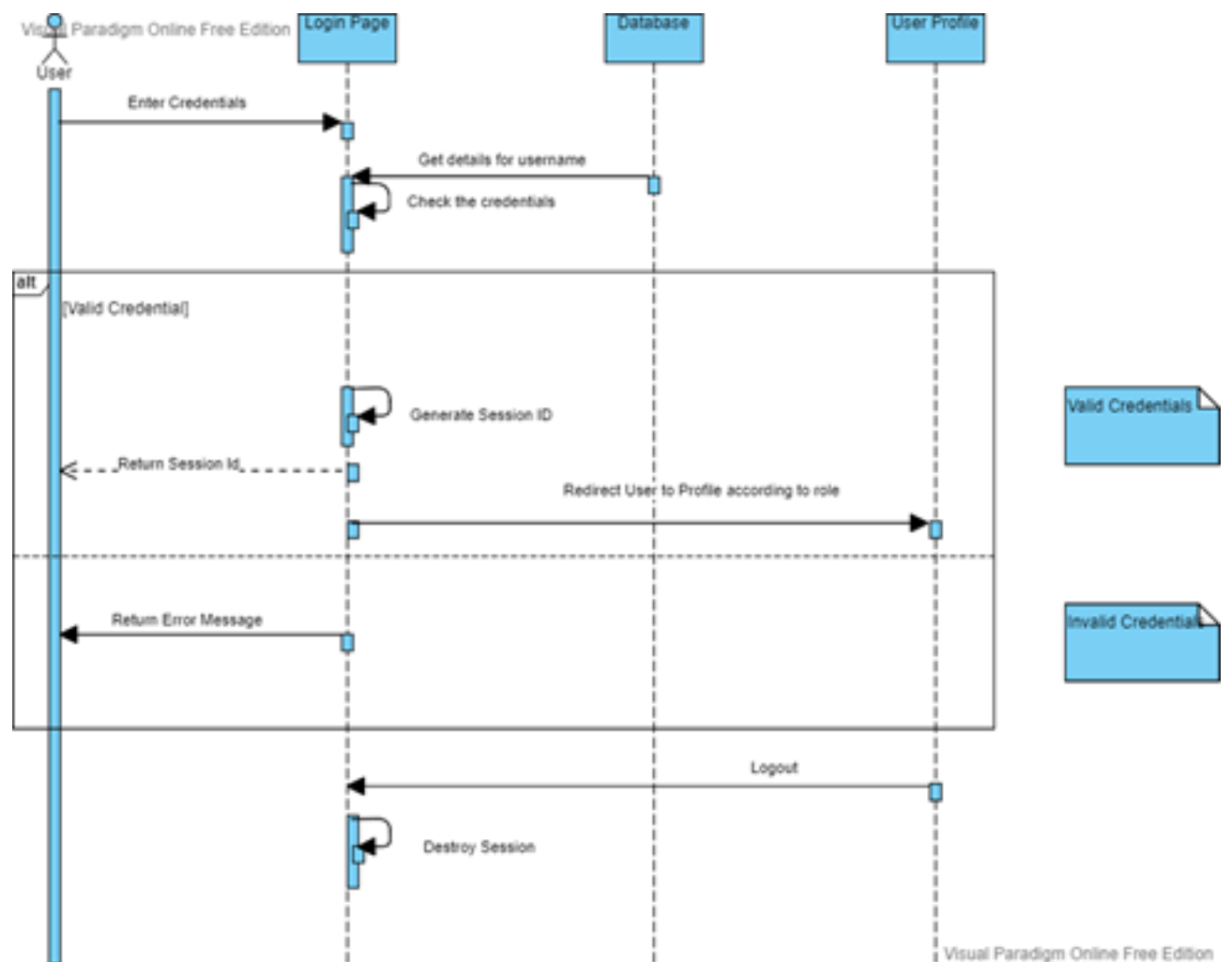


Retailer Activity Diagram

5.4 Sequence Diagram

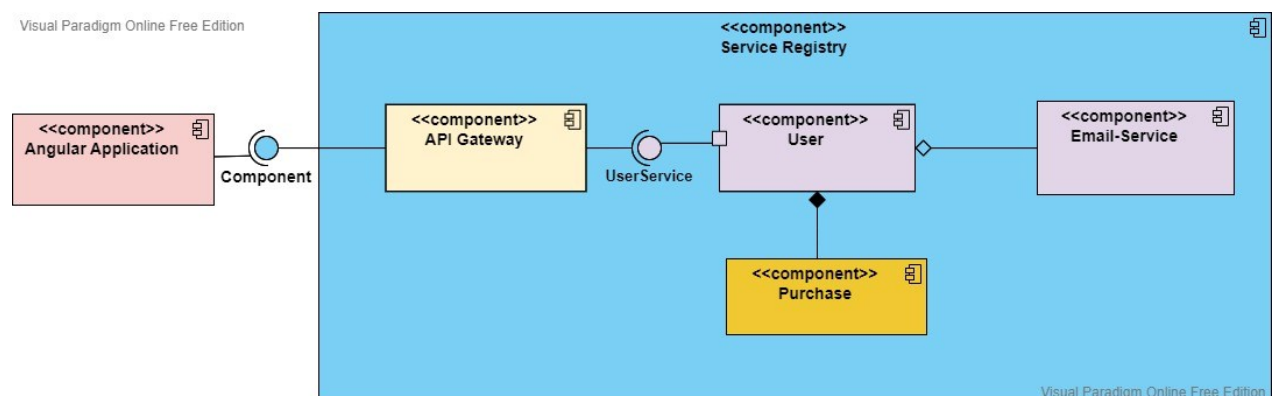
It shows the sequence of flow between activities for specific activities of different actors. The sequence for a user's login and for the retailer purchase is shown in the below diagrams:






5.5 Component Diagram

The component used in our application are as follows:



Chapter-6

6.1 Login Page

 Distribution Management System

DMS Login

Admin Distributor Retailer

Username

Password

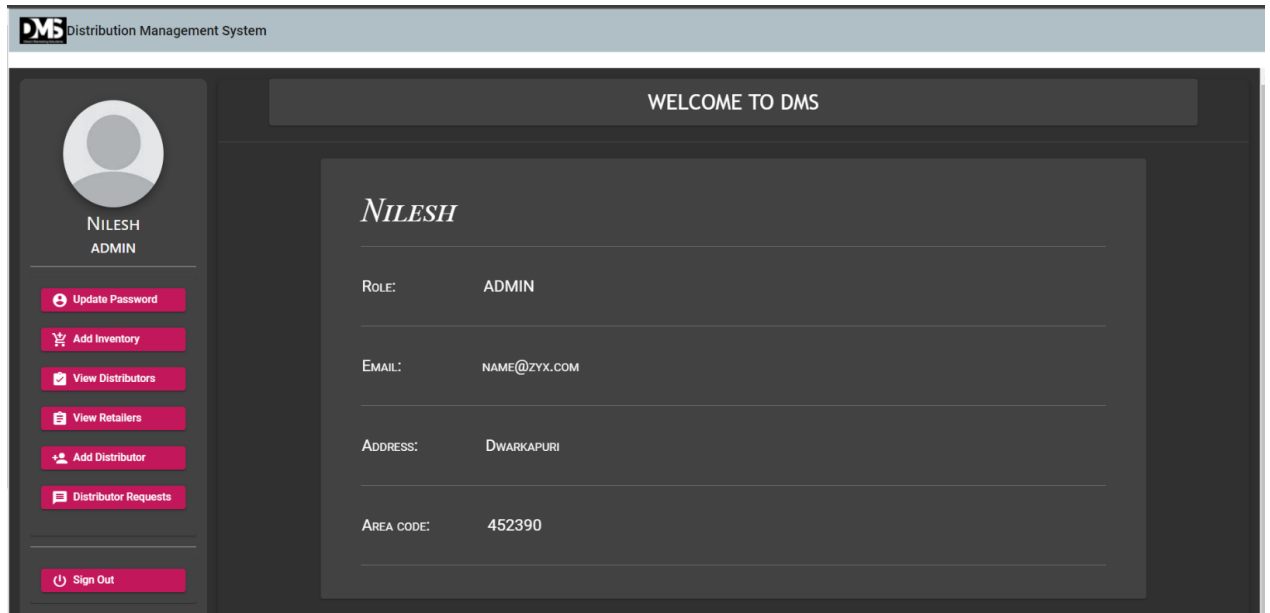
Login

Request Retailer Registration

This is the Distribution Management System Develop by Impetus Technologies

This is the landing page of our Web Application, any intended customer has to have a valid login credentials to get access to other functionalities of the project. The Login credentials include username that is the email id of the user, password and the role he is assigned with which could be chosen from the above three option. If the credentials matches those from the database, the user is routed to its respective Profile page (based on the role) or else is reported with the error message.

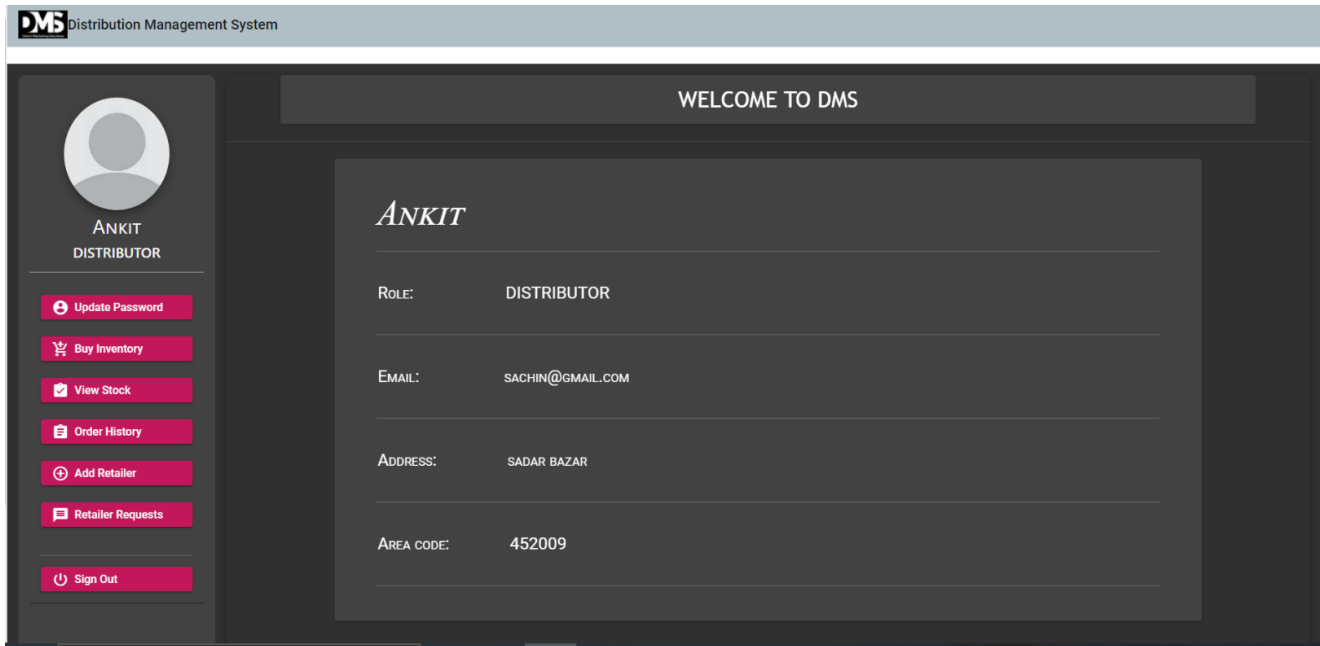
6.2 Admin's User Profile



This is the Admin profile page accessible after the Admin has successfully logged in. This page shows the corresponding details to the user along with a side navigation bar that can be used to serve around various functionalities, which are:

- **Update Password:** User can enter a new password for his account after verifying the old password.
- **Add Inventory:** as our project assumes Admin as the manufacturing unit, therefore admin can view and add new inventories by adding the title, category, description and pricing information about the inventory. Once the inventory is initialized the stock of retailers and distributors are marked as '0' for the corresponding inventory.
- **View Distributor:** Admin can view all the distributors along with their details and the stock they are currently holding.
- **View Retailers:** Admin can also view all the retailers that are currently reporting to the company's distributors.
- **Add Distributor:** Admin can add distributor by entering his name, email, address, area code and a password, once he registers a user the credentials are mailed to his/her registered email id.
- **Distributor Request:** The requests made by distributors of the particular inventory can be viewed using this option.
- **Sign Out:** If user wishes to logout from the application.

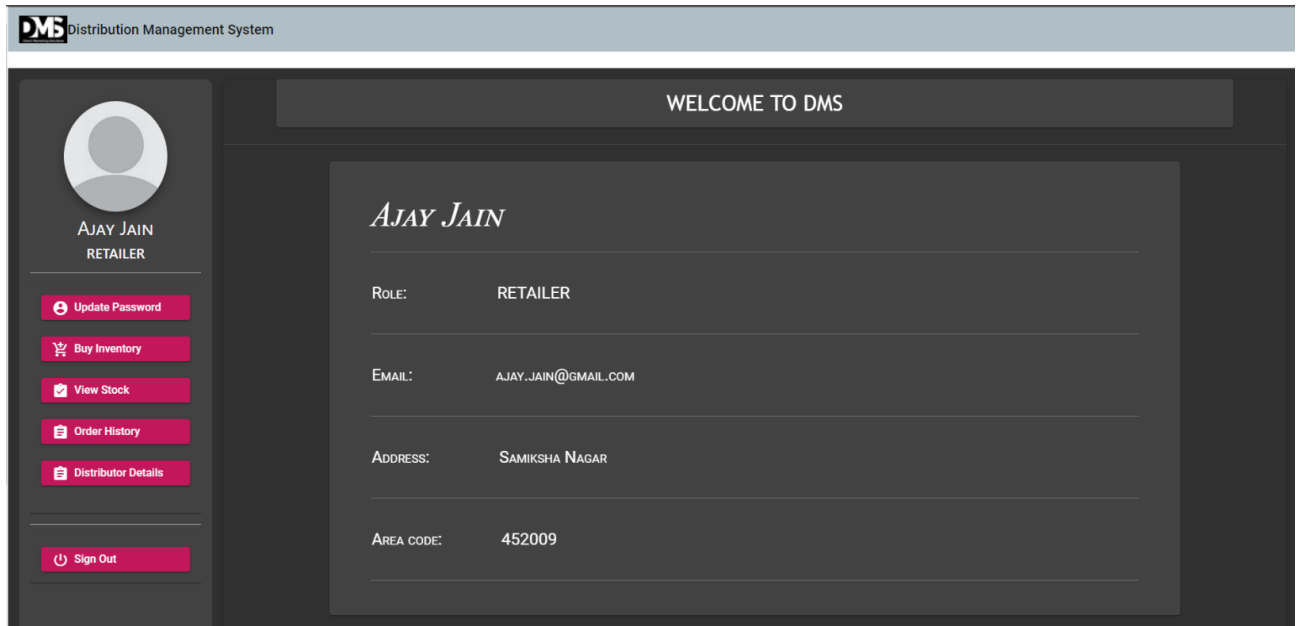
6.3 Distributor's User Profile



This is the Distributor's profile page accessible after the Distributor has successfully logged in. This page shows the corresponding details to the user along with a side navigation bar that can be used to serve around various functionalities, which are:

- **Update Password:** User can enter a new password for his account after verifying the old password.
- **Buy Inventory:** Distributor can view the current available inventory along with its details, further he can click on the buy button to buy an inventory enter the quantity and click on OK. The purchase request for the distributor would be raised.
- **View Stock:** Distributor can view all the stock they are currently holding.
- **Order History:** Distributor can view the details of their previous purchase requests they have raised to admin for stock allocation. They can also view the invoice to corresponding purchase by clicking on the Bill button.
- **Add Retailer:** Distributor can add retailer only for his area code by entering retailer's name, email, address and a password, once he registers a user the credentials are mailed to his/her registered email id.
- **Retailer Requests:** The requests made by retailer of the particular inventory can be viewed using this option; further distributor can allocate the stock to the retailer if he/she has sufficient stock.
- **Sign Out:** If user wishes to logout from the application.

6.4 Retailer's User Profile



This is the Retailer's profile page accessible after the Distributor has successfully logged in. This page shows the corresponding details to the user along with a side navigation bar that can be used to serve around various functionalities, which are:

- **Update Password:** User can enter a new password for his account after verifying the old password.
- **Buy Inventory:** Retailer can view the current available inventory along with its details, further he can click on the buy button to buy an inventory enter the quantity and click on OK. The purchase request for the distributor would be raised.
- **View Stock:** Retailer can view all the stock they are currently holding.
- **Order History:** Retailer can view the details of their previous purchase requests they have raised to distributor for stock allocation. They can also view the invoice to corresponding purchase by clicking on the Bill button.
- **Distributor Details:** Retailer can view details of his/her Distributor that is mapped according to the area code. There can only be one Distributor for a corresponding area code.
- **Sign Out:** If user wishes to logout from the application.

Chapter-7

Summary and Conclusion

Distribution Management System was allotted as a Case Study to me and my project fellows as a part of my training in Impetus Technologies India Pvt. Ltd. Under the leadership and guidance of a wonderful Learning team at Impetus, I had new learnings regarding the technologies I used to build my project. Also, the support and help I received from my amazing mentors and buddies was admirable and appreciable. This project helped me explore new domains in the field of Computer Science.

Distribution Management System was my first project as a web application and quite noticeable one too. I learned about Spring Boot and how helpful it is to develop enterprise level Java applications, using Angular JS to improve user's interface and experience using various resources, components and templates, MySQL as a relational database to manage the data from the application, Git as version control software that helps in resource management and sharing between group members and many more technologies along with the best practices to be considered while developing any professional level project such as using comments between code to maintain the understandability of the code, avoid using hard coded values/properties, handle all possible exceptions, etc.

The project not only helped in building the understanding of various technologies but also importance of planning, discussion before building any project to define the outcomes of your project, build and discuss various scenarios and views for the corresponding project such as logical view, physical view, development view, etc. so the clarity of the project is defined before you start building your project. These are the software development practices that our team followed while building the project.

Chapter-8

Future Scope

The current project satisfies and fulfills the requirements that were presented to us as case study and shows the working of a Distribution Management System in well manner, but as all things can improve, there are some suggested functionalities that could be added in the project:

- An API that will handle the payments of the purchase orders can be added to provide authenticity to the application.
- In current project, we are providing status at only retailer's end that too like a Boolean Yes/No this functionality can further be extended to live order tracking from the end of both buyer and seller.
- This project currently handles only three types of users that are admin, distributors and retailers with no participation of the end customer, so we can propose to add end customer interface for maintain that information on the application too.
- A live messaging section between the adjacent hierarchical members can also be developed to reduce the involvement of any other third party messaging application and maintaining the integrity of the data.
- A feedback system could also be developed which would be directed to the project development team for improving the user experience.

Bibliography

- <https://angular.io/docs>
- <https://dev.mysql.com/doc/>
- <https://spring.io/projects/spring-boot#learn>
- <https://www.eclipse.org/downloads/>
- <https://code.visualstudio.com/docs>
- <https://material.angular.io/guide/getting-started>
- <https://www.apptivo.com/solutions/distribution-management/>
- <https://www.arokiait.com/>
- <https://online.visual-paradigm.com/app/diagrams/>
- <https://git-scm.com/doc>