CUSTOMER DATA MANAGEMENT

Fastenal India Sourcing IT & Procurement Pvt Ltd

Bangalore

FINAL DISSERTATION REPORT
SUBMITTED TO MANIPAL ACADEMY OF HIGHER EDUCATION, MANIPAL

In partial fulfilment of the requirements for the award of degree of

MASTER OF COMPUTER APPLICATIONS

By
Abhinandan Mishra (Reg. No: 210970061)

Under the guidance of

Dr Nandini Kumari
Assistant Professor
Department of Data Science and
Computer Application

Vamsi Krishna Reddy Pallamala
IT Associate Manager
Fastenal India

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Department of Data Science and Computer Application

May 2023

CERTIFICATE

This is to certify that the project titled **"CUSTOMER DATA MANAGEMENT SYSTEM"** is a record of the bonafide work done by **Abhinandan Mishra (210970061)** submitted in partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications (MCA) in **Department of Data Science and Computer Applications** of Manipal Institute of Technology, Manipal, Karnataka, (A Constituent Unit of Manipal Academy of Higher Education), during the academic year 2022 - 2023

Dr. Nandini Kumari
Assistant Professor

Department of Data Science &
Computer Applications

P.Vamsi krishna Reddy

Vamsi Krishna Reddy Pallamala
IT Associate Manager

Fastenal India

Dr. Radhika M Pai
HOD and Professor

Department of Data Science &
Computer Applications

Internship Completion Certificate

ABSTRACT

The project – Fastenal: Customer Data Management, is a proof-of-concept project, assigned by Branch Sales – Team of Fastenal India Sourcing, IT & Procurement Pvt Ltd. This project deals with listing of Customers and Accounts associated with the Specific Customer of Fastenal and plots all the locations in a map. This proof-of-concept project is aimed to test our technical skills from the company on Angular, ASP.NET 6, Angular Material, MS-SQL, and Azure.

To start with, we documented High-Level Design and Low-Level Design Documents. After the review of both, a GitHub repository was created, which was used as the Version Control System for the web application. After that we divided the work among us – created separate git branches and started with the backend of the application, which was a web API. We merged our codes from time – time, for updates. After code review of it, we went ahead with the front-end of the application. Finally, the application was reviewed by both our team lead and solution architect, which was then presented to them, with explanation of the web app. Finally, we merged the code to master git branch.

We were successfully able to achieve the goal of the project – showing Customers & Accounts in a table form and as a cluster of markers in a google map to the customers. We also implemented server-side pagination (without sorting and filtering options), but our app can sort and filter in a particular page. Inside google-map, to access all the features of the app, one must login to the application and a feature to change password is also included with the app for logged in users.

In conclusion, we were able to achieve the objectives defined by the solution architect of branch sales, thus successfully completing the project. User Interface of the application was solely made using Angular Material. The tech stack of the project – Angular 13 for frontend and .NET 6 - Web API for backend, and MS SQL as the database. We also used Postman and Swagger UI for API-Testing purposes and hosted on Azure.

ACKNOWLEDGEMENT

First, I would like to thank my external guide, **Vamsi Krishna Reddy Pallamala – IT Manager, Fastenal, India**, for providing appropriate suggestion from time to time, encouraging me throughout the internship, to achieve more in the project.

Next, I would also like to thank **Dr Radhika M Pai – HOD and Professor, Department of Data Science and Computer Applications**, for her constant support and guidance during the period of my internship project work.

Next, I would also like to thank **Dr Nandini Kumari – Assistant Professor, Department of Data Science and Computer Applications**, for her constant support and guidance during the period of my internship project work.

Next, I would also like to thank **Sidharth Telkar – Solution Architect, Branch Sales, Fastenal India**, for doing the code review time to time, teach concepts, and provide useful and appropriate feedback which made the final project successful.

I would also extend my thanks to **Abhishek Gugulla – Developer, Branch Sales, Fastenal India**, for guiding me throughout the internship with trainings required, and providing appropriate feedbacks, which helped me to work with Vigor till the end.

Finally, I take this opportunity to extend our earnest gratitude and respect to **my parents, my colleagues and batchmates** for their direct and indirect support during the period of my internship project work. I will remain in debt to my alma mater for making me what I am.

Place: Bangalore **Abhinandan Mishra**

Date: May 15,2023 **210970061**

CONTENTS

Table of Contents			
			Page No
Abstract			iv
Certificate			v
Acknowledgement			vi
List Of Figures			viii
List of Tables			ix
Chapter 1		**INTRODUCTION**	**1**
	1.1	Introduction	1
	1.2	Area of work – General Discussion	1
	1.3	Project Schedule	2
	1.4	Organization of the Project Report	3
Chapter 2		**BACKGROUND THEORY**	**4**
	2.1	Introduction to Project Title	4
	2.2	Exploring Tech Stack	5
	2.3	Exploring Software Tools	7
	2.4	Conclusion	9
Chapter 3		**METHODOLOGY**	**10**
	3.1	Introduction	10
	3.2	High – Level Design	10
	3.3	Components Design	18
	3.4	Exploring the Web Application	26
	3.5	Conclusion	32
Chapter 4		**RESULT ANALYSIS**	**33**
	4.1	Introduction	33
	4.2	Result Analysis – Web Application	33
	4.3	Significance of Results Obtained	41
	4.4	Conclusion	41
Chapter 5		**CONCLUSION AND FUTURE SCOPE**	**42**
	5.1	Brief Summary of Work	42
	5.2	Future Scope	42
REFERENCES			**44**
PROJECT DETAILS			**45**

LIST OF FIGURES

Figure No	Figure Title	Page No
3.1	Process Flow Diagram	12
3.2	Sequence Diagram	13
3.3	Use Case Diagram	15
3.4	Class Diagram	16
3.5	Data Design	25
3.6	Architecture Diagram	32
4.1	Register Screen	33
4.2	Login Screen	34
4.3	Customer Dashboard	34
4.4	Add Account Form	35
4.5	Update Customer Form	35
4.6	Delete Customer	36
4.7	Account Home	36
4.8	Add Account	37
4.9	Update Account	37
4.10	Account Plot	38
4.11	Account Details	38
4.12	Navbar	39
4.13	Logs	39
4.14	Reset Password	40
4.15	Search Bar	40

LIST OF TABLES

Table No	Table Title	Page No
1	Project Schedule	3
2	API Catalogue	24
3	Project Details	45

Chapter 1
Introduction

1.1 Introduction to the company

Fastenal is a company that is different to many different customers: a logistics company, a technology provider, however, in a more general case, a distributor of wide-ranging industrial and hardware products. All these aspects of the service provided by the company share a common foundation: 'Growth through Customer Service'. It is a Winona, Minnesota-based American corporation. Based on its 2020 revenues, it was placed 479 in the Fortune 500 for 2021, with service model centers in about 3,200 in-market locations.

Fastenal India IT, Sourcing and Procurement Pvt. Ltd was founded in Bengaluru, Karnataka, in September 2013 with the goal of creating an Indian Technology Centre of Excellence. As a member of Fastenal's IT team, we share the same engagement and obligations in IT initiatives as the corporate office

Branch Sales team at Fastenal has previously developed many applications which includes a Point of Sales desktop application, Web POS application etc. The project assigned to us, is to design and develop a customer data management system, to store golden records of a customer and display the location of the accounts on a map.

The areas of computer science covered in this project include in-house development of software applications for Store Solutions, i.e., .NET application development, Database management Systems, Web development and API (Application program Interface) development.

1.2 Area of work – General Discussion

This document gives the technical overview of the "Fastenal: Customer Data Management" web app. All its functionalities, including various components used, basic navigation flow, sequence diagram, data design showing the schemas used and API catalogue to fulfill the intended features, have been discussed in the document. This serves as a technical document and may be used by any concerned parties to understand the functioning as well as the development roadmap of the forementioned web application.

NOTE: Due to the NDA (Non – Disclosure Agreement) signed by me, a few pieces of information like data used as well as all the codes written has not been revealed. Also, services and terms internal to Fastenal India IT, Sourcing and Procurement Pvt. Ltd have been explainedminimally.

While developing the project, we have followed the SDLC processes closely. This project which is a standalone web application uses most of the technologies used in branch sales team's real projects.
This project is just a proof-of-concept project, done to evaluate my technical skills on various technologies used inside the company, but a similar project, with similar objectives is in development, and it would help the organization in managing of customers and associated account. So, it'llbe easy for organization to know details about all the Customers and Accounts of their state, which would ease searching and navigation easier.

The objective of this project is to design and develop a customer data management system, to store golden record of a customer. A user would be able to access the dashboard to view and update the customer data. Ability to add, edit or delete accounts for a given customer. Finally, by plotting the accounts' locations in a map, we give the user an overview of where and how close are the accounts in holistic view. Also, there are some constraints that need to fulfill:

- Demonstration of uniqueness of customer record.
- Demonstration of uniqueness of account linked to a customer.
- Demonstration of the application that it is hosted on a cloud platform

1.3 Project Schedule

Note: This schedule starts just after the training period at Fastenal. The training period schedule is included in the mid-term report. Week Starts from April 3rd. It's marked as the 14th Week after my internship started at Fastenal

Topics	
Week 14	Discussion of Project Objectives
	Documentation of High-Level Design Document
Week 15	Review and approval of High-Level Design Document
	Discussion for Low Level Design Documentation
Week 16 and 17	Creation and Completion of Low-LevelDesign Document with multiple reviews following it.
Week 18 and 19 (Mid)	Starting of the API and creation of database of the application
	Completing the API with required action methods.
	Completed final review of the API.
Week 18 (Mid) and 19	Started creating Front-End UI components
	Completion of all the Front-End UIComponents and integration of both API and front-end.
Week 21	Final Review and presentation of the project to supervisors of our team.

Table 1 – Project Schedule

1.4 Organization of the project report

Chapter 2 includes information on various technologies used in the web application and an overall overview on how all technologies work together in a combined fashion.
Chapter 3 includes the methodology of how the app was developed and contains a detailed explanation on how all the components of the app were designed and worked on.

Chapter 4 includes the screenshots of the results obtained a brief discussion on the same.
Chapter 5 discusses the conclusion of the project and things else can improved in the web application.

Chapter 2

Background Theory

2.1 Introduction to Project Title

Customer Data Management is a web application that provides an interface to the user (Admin) for loggingin and viewing Customers details and Accounts associated with the customer and their locations on map.

It provides a platform for the authorized users to login using their credentials. There is real time client-side validation. On successful login, it redirects the user to the Customer Dashboard where user can add new customer, update existing customer and also delete the customer, also user can view the details of the customers from the list and when user will click on the specific customer it will redirected to the Account dashboard of that customer where user can add, update and delete the accounts for that specific customer and also able to check the locations of the accounts on map.

Dashboard has the options to reset password and logout. Logout makes the current user observable null and redirects to the login page.

Reset password function sends a PATCH request to API that replaces the current password with new password along with the client-side validation.
Get customer list uses tables in Angular Material Component to display the list as well as add features like sort and filter.
Map view uses angular's native google maps component to plot the regions based on the latitude and longitude data, and markers marking the regions have an animation embedded and when clicked it shows the branch code and city in a map-info view pane.

Various functions that are used are login, logout, change password, get branch list and get branches in map. The application uses Angular as frontend hosted on .NET platform, API layer between client and the sink and MS SQL as it's sink.

2.2 Exploring the Tech Stack

2.2.1 Hypertext Markup Language – HTML

HTML, or Hypertext Markup Language, is the industry standard markup language for web-based authoring. Both CSS and programming languages like JavaScript can be useful.

HTML files from a web server or locally saved files are converted into multimedia web pages by web browsers. In its early iterations, HTML featured visual indicators for the document's presentation and logically represented the structure of a web page.

HTML permits scripting languages such as JavaScript to contain programs that alter the appearance and content of web pages. The appearance and layout of content is controlled by CSS. The World Wide Web Consortium (W3C), which previously maintained HTML standards but now manages CSS standards, has encouraged the usage of CSS over explicit presentational HTML since 1997.

2.2.2 Cascading Style Sheets - CSS

CSS is a style sheet language for describing the appearance of a document written in a markup language such as HTML. CSS, like HTML and JavaScript, is an important part of the Internet. CSS is a style sheet that separates appearance from text in terms of layout, colors, and fonts. By defining suitable CSS in a separate file, this separation can improve content accessibility, enable more freedom and control when setting presentation characteristics, and allow numerous web pages to share formatting.
Allow the CSS file to be cached to improve page load time between pages that share the file and its formatting, reducing structural content complexity and duplication.

2.2.3 Typescript

The TypeScript programming language was created and is maintained by Microsoft. It's a syntactical superset of JavaScript with static typing support. It's designed for large-scale applications and JavaScript conversion. Because TypeScript is a superset of JavaScript, existing JavaScript programs are also valid TypeScript programs.

Both client-side and server-side JavaScript applications can be built with TypeScript (as with Node.js or Deno). Transpilation can take several different forms. The TypeScript Checker or the Babel compiler can be used to convert TypeScript to JavaScript.
There was a demand for bespoke tooling to make developing JavaScript components easier due to the complexity of dealing with large amounts of JavaScript code.

2.2.4 NET core

.NET core is a free and open-source managed computer software platform for Windows, Linux, and macOS. It is a cross-platform framework that replaces the.NET Framework. The project is mostly developed by Microsoft personnel and provided by the.NET Foundation under the MIT License.

The .NET Framework supports ASP.NET Core online apps, command-line/console programs, libraries, and Universal Windows Platform apps. Windows Forms and Windows Presentation Foundation (WPF), which provide the basic user interface for Windows desktop apps, were not implemented prior to.NET Core 3.0.

NuGet packages are supported by the.NET Framework. Unlike the.NET Framework, which is updated through Windows Update, it receives updates through its package manager.

2.2.5 MS SQL

The relational database management system (RDBMS) known as MS SQL (Microsoft SQL) was created by Microsoft. Users may conveniently and securely store, retrieve, and manipulate data with this robust and well-liked database management system.

MS SQL Server supports several programming languages and frameworks, including as SQL, .NET, C#, and Java, and it can be utilized with a variety of platforms, including Windows, Linux, and Docker.
Database administrators and developers may take use of a variety of capabilities and tools offered by MS SQL Server, including a query optimizer, indexing, transaction processing, security, backup and recovery, and more. In-memory computation, column store indexing, and

real-time operational analytics are further sophisticated features available.

2.2.6 Angular

Google's Angular Team and a community of individuals and organizations produced Angular (also known as "Angular 2+" or "Angular CLI"), a TypeScript-based free and open-source web application framework. The AngularJS framework was fully redesigned by the same team that built AngularJS.

Angular is a set of technologies that aid in the construction of the greatest foundation for our apps. It's entirely configurable and cross-platform compatible. Any feature can be altered or removed to match our development methodologies and feature needs.

2.3 Exploring Software Tools

2.3.1 Microsoft Visual Studio

Microsoft Visual Studio is a Microsoft-developed integrated development environment (IDE). It's used to make websites, web apps, web services, and mobile apps, among other things. Windows API, Windows Forms, Windows Presentation Foundation, Windows Store, and Microsoft Silverlight are among the Microsoft software development platforms used by Visual Studio. It can generate native as well as managed code.

The code editor and debugger in Visual Studio may work with almost any programming language if a language-specific service is available (to varied degrees). Visual Studio currently supports 36 programming languages. Among the built-in languages are C, C++, C++/CLI, Visual Basic.NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML, and CSS. For other languages including Python, Ruby, Node.js, and M, plug-ins are accessible. Java (and J#) was supported in the past.
Visual Studio Community Edition is the most basic and is available for free. The Visual Studio Community edition's motto is "Free, fully featured IDE for students, open-source, and independent developers."

2.3.2 Postman

Postman is an API creation, testing, and iteration platform for developers. As of April 2022, Postman claims to have over 20 million registered users and 75,000 open APIs, making it the largest public API hub in the world. The company's headquarters are in San Francisco, but it is also based in Bangalore, where it began.

The following are the numerous goods available:

API repository: Users can save, categorize, and collaborate on API artefacts in public, private, or partner networks using this central platform.

API builder: Facilitates the creation of an API design pipeline using Open API. Various source controls, continuous integration, and delivery (CI/CD),gateways, and APM systems are all integrated.

API client: API design, API documentation, API testing, mock servers, and API detection are just a few of the accessible tools.

Intelligence: Examples of intelligence include security alerts, API repository search, workspaces, reporting, and API governance.

Workspaces: Using public, private, and partner workspaces, developers can collaborate both inside and outside the company.

2.3.3 Swagger UI

Swagger UI is a tool used to visualize and interact with the API resources. It is automatically included in the project with all the configuration settings done by OpenAPI in Visual Studio while creation of the API. It makes life easier by allowing us to test the API endpoints with a visual UI and helps us to debug them by showing what went wrong in the error section of the UI.

It is Dependency Free, which means the UI will work in any dev environment. It allows developers to effortlessly interact all the operations that the API exposes for easy consumption. It is also fully customizable which means one can tweak the UI and functionality as they require for the project.

While creating a visual studio project, one just must select the option to include OpenAPI support in-order for the configuration to be configured in the project.

2.3.4 GitHub Desktop

This is the desktop app of GitHub, which acts as an alternate to the GIT CLI and allows us to clone / fork a repository with the help of the URL.

Once that is done, a folder is created in the local machine, where the whole project is located, and once changes are continuously made in the app files, it gets reflected in the GitHub desktop. This can then be committed and pushed to origin, which means that it will be visible to all others who are part of the repository, and the code will be available for viewing or merging with the other branches as and when required.

Since GitHub is a version control system, one can easily merge, revert to a particular commit – which means if something goes wrong with the current stage of the application, one can easily go back to the working version of the app, which makes life easier for a software developer.

Once all the development and code-review are completed, a pull request can be made for merging into the master branch, which will then be done by the owner of the repository.

2.4 Conclusion

This chapter discusses on various fronts about the background theory of the project, the tech stack used, and the software tools used to complete the web application successfully.

Chapter 3

Methodology

3.1 Introduction

This chapter explains about the high-level design document, which includes the flow diagram of the web application and sequence diagram of the web application. It also includes the data design from the low-level design document. It also contains the explanation for why database was designed in a particular way using MS SQL.

It also briefly explains the angular application, its components, then explanation of API is also included along with them.

3.2 High Level Design

3.2.1 Product Perspective

The Fastenal: Customer Data Management will be made up of several different components. Some of these components will be programmed, while others will be implementations of open-source programs. The administrative and user interfaces will be using Angular as the front end, hosted on the .NET platform to display the pages, and MS SQL to retrieve, insert, delete, and update the database. It will also be using the Angular Material Component Library.

3.2.2 Tools Used

- **Angular 15:** Angular is a TypeScript-based free and open-source web application framework led by the Angular Team at Google and by a community of individuals and corporations.

- **Angular Material Component Library:** Angular Material is a User Interface (UI) component library that developers can use in their Angular projects to speed up the development of elegant and consistent user interfaces.

- **.NET 6**
- **MS-SQL: A Relational database**
- **GitHub: Version Control System**

3.2.3 General Constraints

The Fastenal: Customer Data Management System is a user-friendly application and is as automated as possible. The users can see the login page only after login. After logging in, users can view the location of the accounts of a customer on a map.

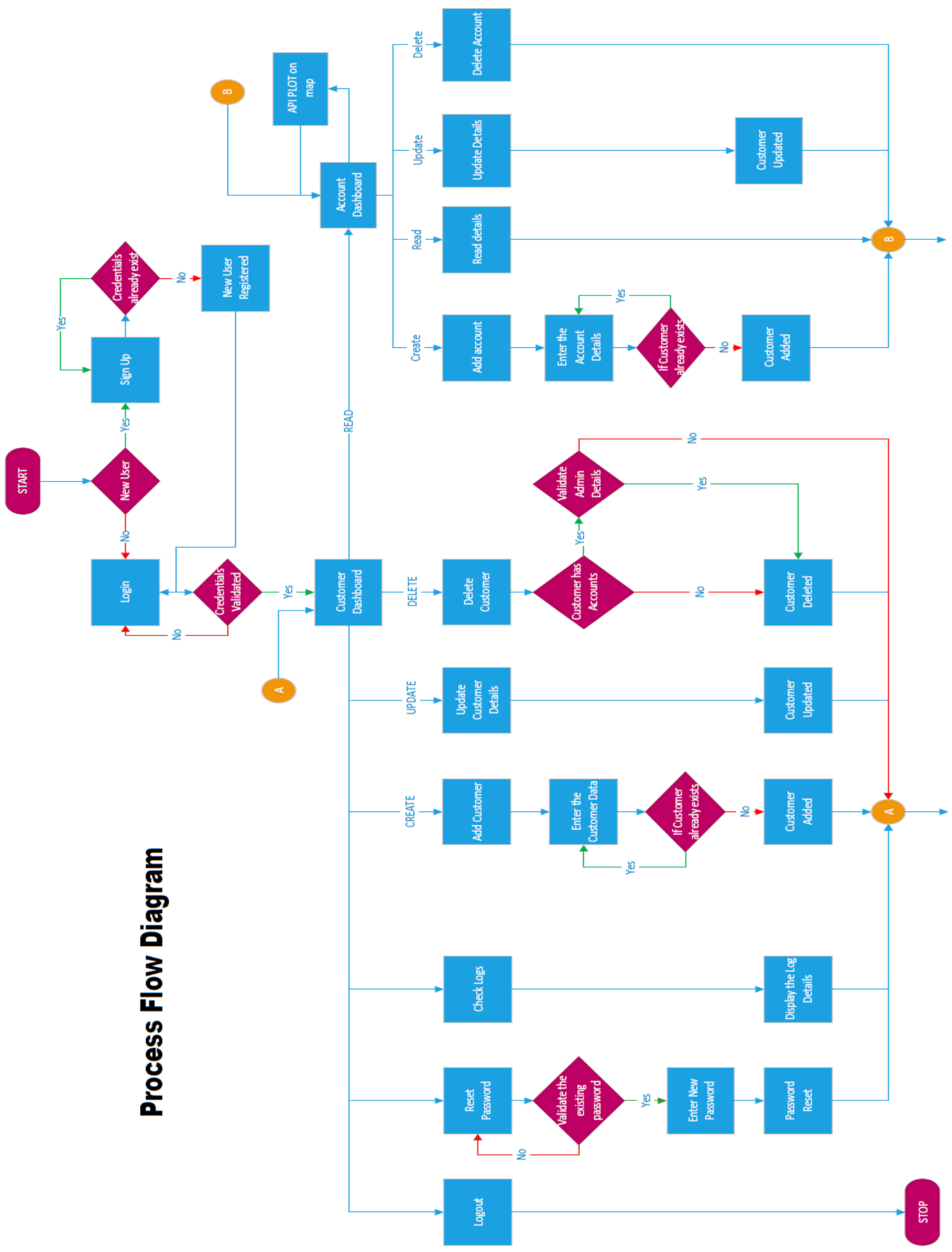
3.2.4 System Design

3.2.4.1 Main Design Features

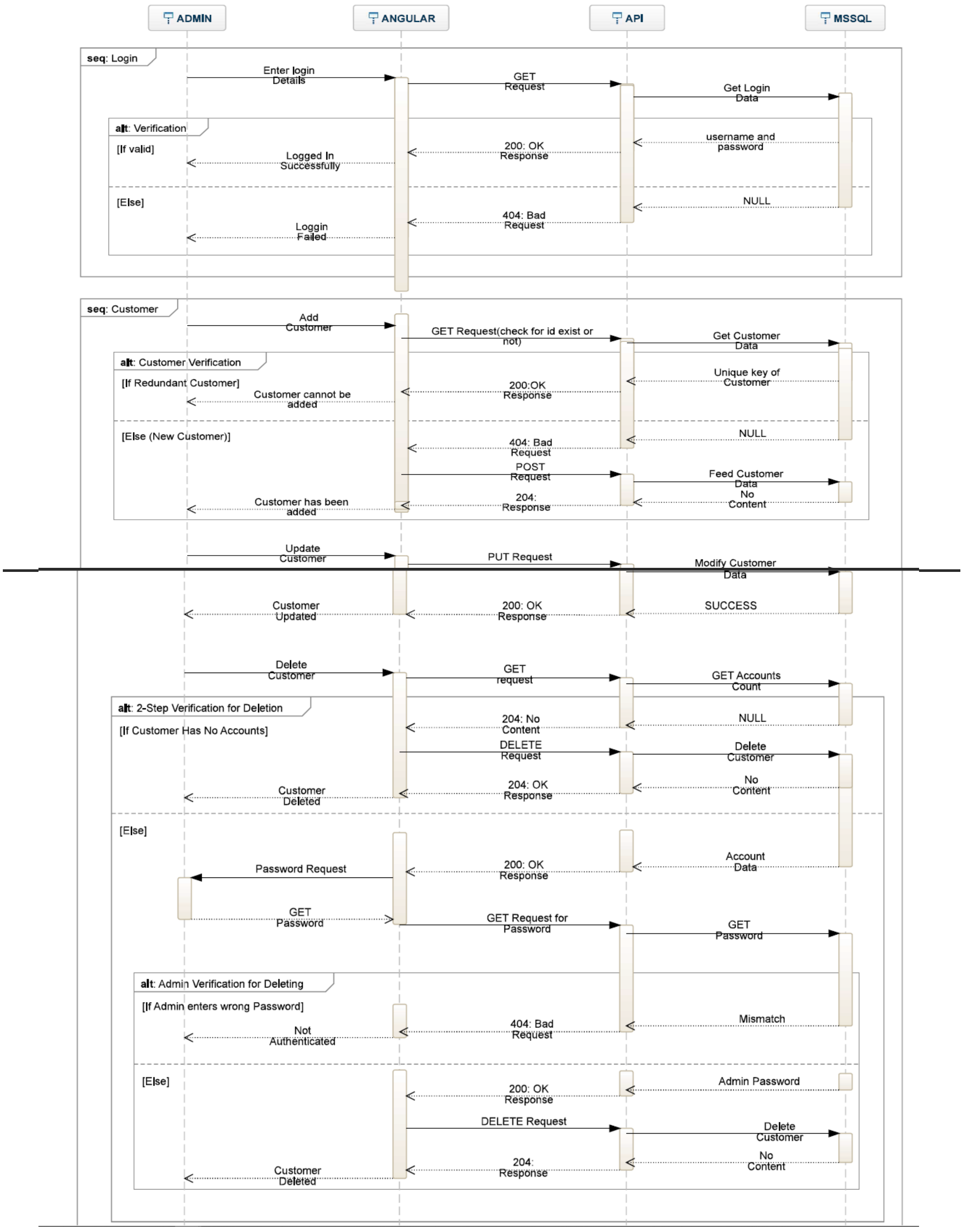
The main design feature includes four major parts: the architecture, the user interface design, designing of API (Application Programming Interface) layer between the client and the sink and the database design. To make these designs easier to understand, the design has been illustrated in attached diagrams.

3.2.4.2 Process Flow Diagram

Figure 3.1 - Process Flow Diagram



3.2.4.3 Sequence Diagram



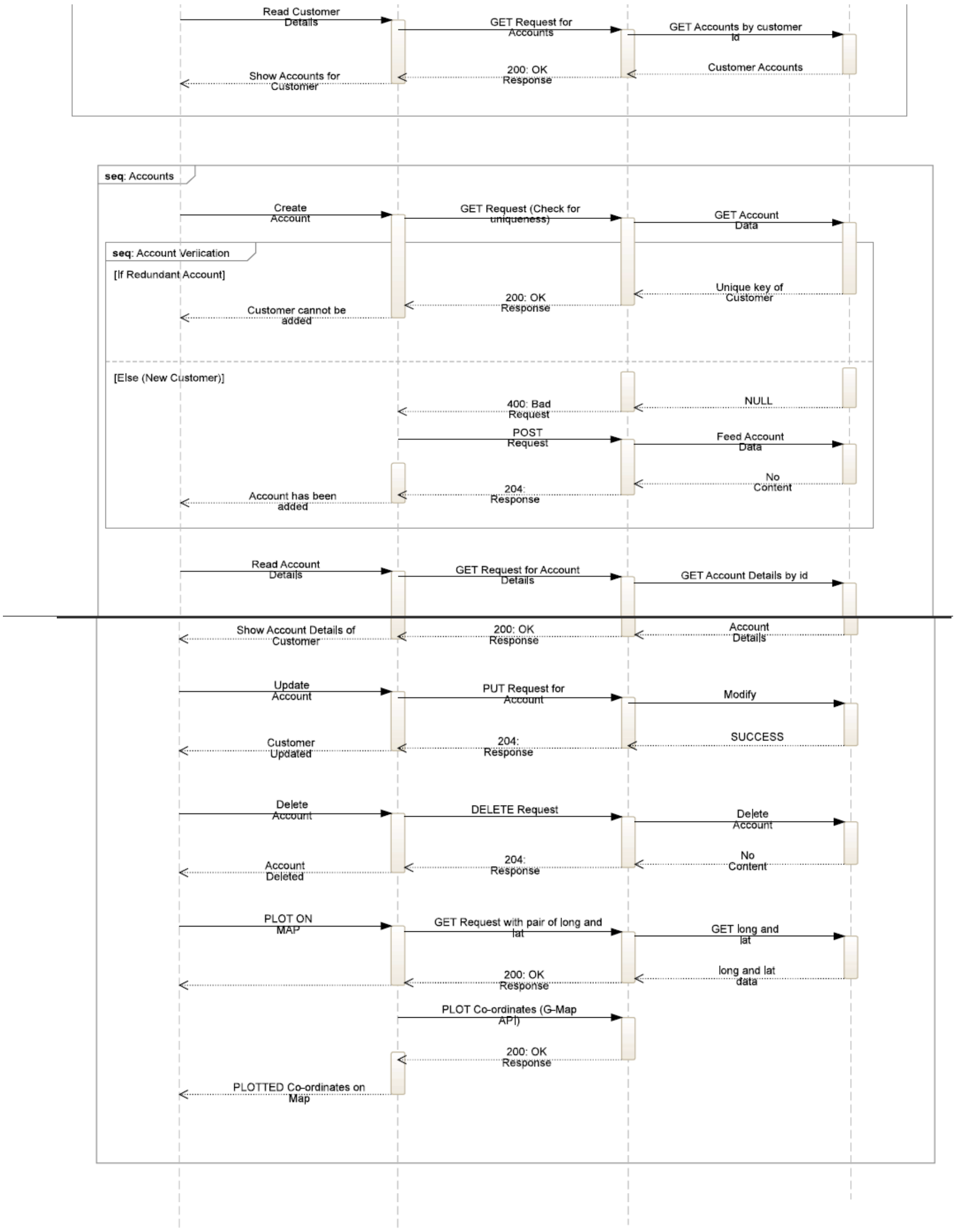


Figure 3.2 - Sequence Diagram

3.2.4.4 Use Case Diagram

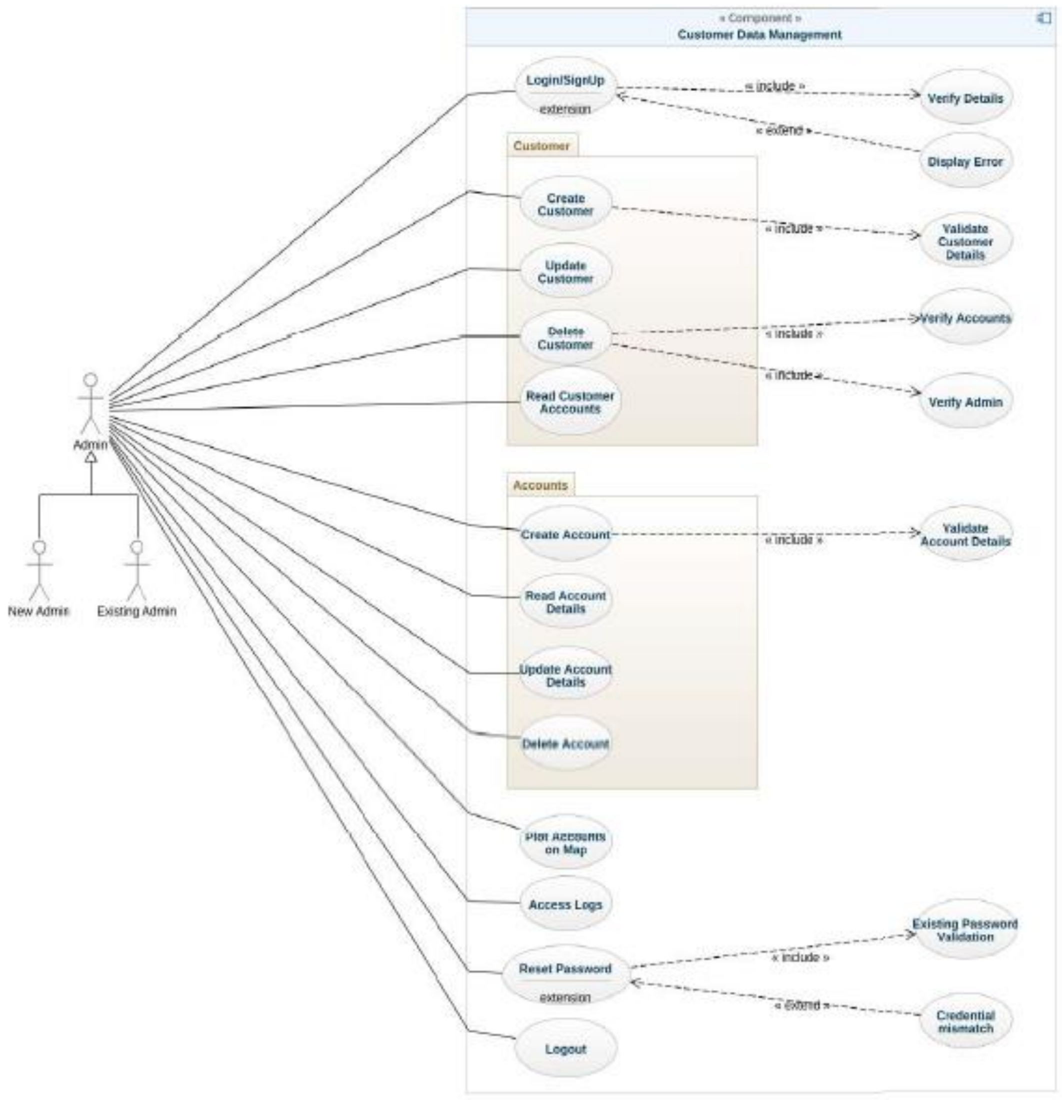


Figure 3.3 – Use Case Diagram

3.2.4.5 Class Diagram

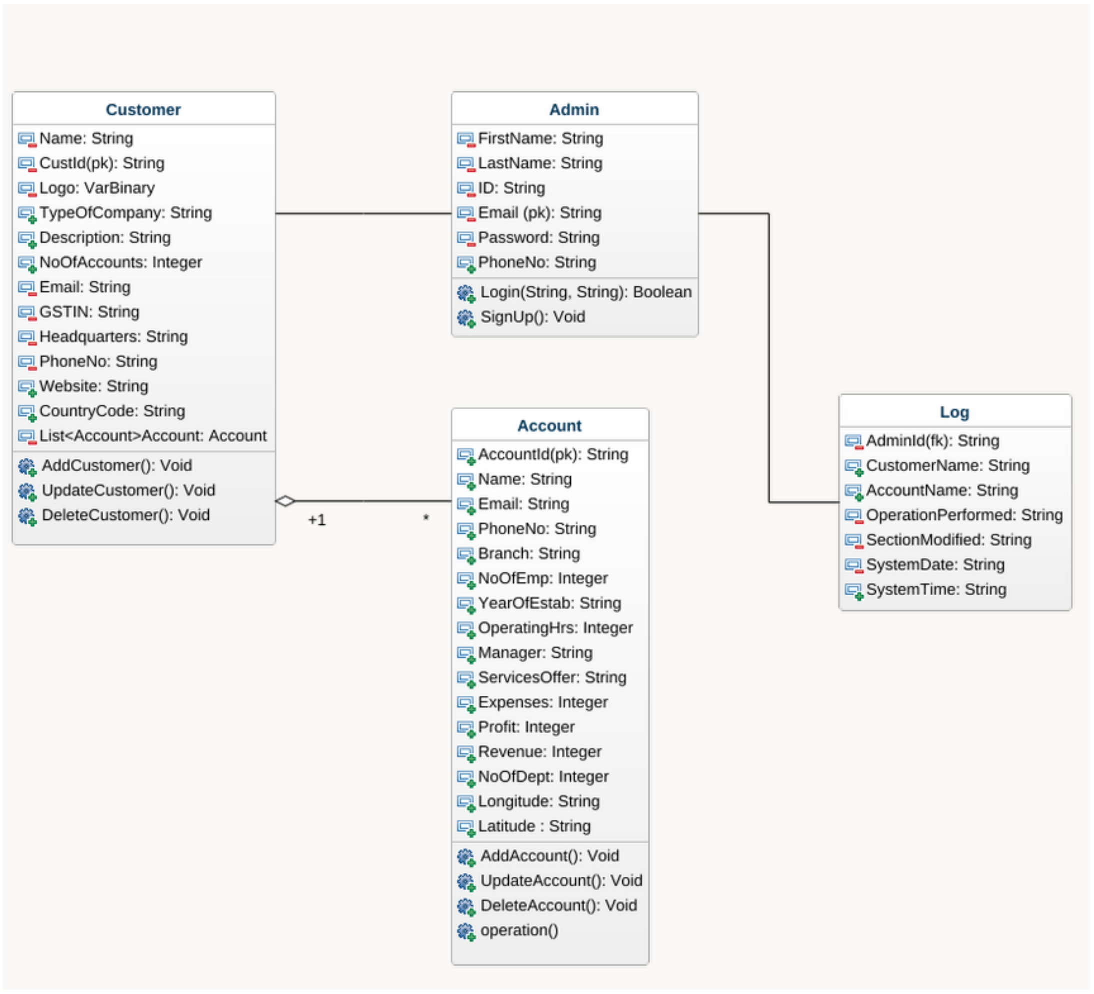


Figure 3.4 - Class Diagram

3.2.5 System Overview

This project is a standalone web application. Authentication is required before proceeding to the dashboard. The users will only be able to see the login page until they log in. After logging in, users can view the customer records and their accounts along with the location. The login credentials stored within the database are not stored as plain text, they are stored after encryption.

The user is also allowed to reset/change his password. User can view the location of the accounts of a customer which is displayed on dashboard and can also filter the data by every column mentioned in the list.

The database will not be made accessible to any users, it will only be accessible by administrators.

The user interface is a simple plain layout with little graphics designed using Angular Material Component Library. It will display information very clearly for the user and will primarily output information to the user through HTML pages.

3.2.6 Security

Security is not the prime focus of this project, only the minimal aspects of security will be implemented. The user credentials would be encrypted before getting stored in the database.

3.2.7 System Overview Introduction

The front end of the program is a web application. Authentication is required before proceeding to the dashboard. After logging in, users can view the accounts of each customer. The login credentials stored within the database are not stored as plain text, they are stored after encryption.

The database will not be made accessible to any users, it will only be accessible by administrators.

The user interface is a simple plain layout with little graphics designed using Angular Material Component Library. It will display information very clearly for the user and will primarily output information to the user through HTML pages.

3.2.8 Client-Side Validation

In our project, at three specific places we are performing client-side validation.

The first place where we perform validation is the login page. For login, one needs to input their username and password. The username is the email address, and before using it for authentication purposes, we validate it at the client-end, using the in-built feature available in forms module. We also make sure that any of the fields in login page is not empty and as said above, that the username is an email address before sending for authentication.

The second place where we check is while resetting the password. We confirm the entered password to check its strength, and ask the user to enter the password again, and check the equality of the two, before encryption and storing it to the database.

The final place where a validation is performed is to check whether any of the retrieved data is null or not, so that the app doesn’t break in the middle of execution. We would also implement some logic to make sure that the app doesn’t break if null is returned.

3.3 Components Design

3.3.1 Data Classes

Customer: The Customer class represents a customer entity in a business system. It typically contains information such as customer's name, description, address, sector, email, and size. This class includes methods to manage customer-related functionalities, such as adding, updating, and deleting customers.

Admin: The Admin class represents a user entity in a system, usually associated with authentication and authorization. It typically contains information such as username, password, and email.

Logs: The logs class represents a historical record of actions or events that occurred within a system. It stores information such as timestamps, user actions which were already performed.

Account: The Account class represents a customer's account in a system. It includes methods for managing user account information. This class also includes methods to manage account-related functionalities, such as adding, updating, and deleting accounts.

3.3.2 Template classes/components:

Dashboard Component: The Dashboard Component class represents a graphical user interface (GUI) component that displays relevant information and statistics on a dashboard. It includes methods for retrieving and processing data from various sources, such as databases or APIs, and presenting them in a visually appealing and informative manner. There are two subcomponents in the Dashboard Component, the customer dashboard and the account dashboard component. Once the user logins into the application, he/she is redirected to the customer dashboard where he/she can create, view, update and delete the customers. Upon clicking on a customer, he/she is redirected to the accounts dashboard. The account dashboard allows the user to create, view, update or delete the accounts.
A button to view the locations of the accounts of a customer is also present in the customer dashboard.

Header - This component is the header of the whole project and contains links to few of the socials of the company.

Footer - This is the header component of the whole project, and this contains few of the functionality for the page like searching and logging options.

Map Component – This component contains the functionality for the account to be plotted on the map using the Angular Google Maps and API call.

Login: The Login class handles the process of user authentication and login in a system. It includes methods for validating user credentials, checking user permissions, and managing user sessions.

3.3.3 Angular Component & Functions

Login

login (Username, Password) – On clicking the Login button, this function will be called which will pass on the Username and Password entered by the user to login(params) function in AUTH Service.

getAdmin() – This function is used to get the details of a particular admin. This function also sets values of admin name and admin email into local storage.

Register:

register () – This function is used to register a new admin. It validates the form and posts the values result into the database.

Header:

addcustomer() – This function is used to open the dialog of the add customer component.

addaccount() - This function is used to open the dialog of the add customer component.

logout() – This function is used to logout the current logged in user and redirects them to the login page.

onSearchTextChanged() – This function is used to implement search bar.

clearSearch() – This function is called when the focus is out of the search bar.

resetPassword() - This function is used to open the dialog of the reset customer component.

Footer:

Reset Password:

resetPassword() – This function is used to put the new values of the password into the database.

close() – This function is used to close the dialog box.

Customer Home:

getList() – This function is used to get list of the customers.

editCustomer() – This function is used to open the dialog box to edit the customer information is the database.

getCustomerName() – This function is used to get the name of the customer name.

deleteCustomer() – This function is used to delete the customer.

Add-Customer:

addCustomer() – This function is used to add a customer or update a customer.

closePopup() – This function is used to close the popup.

Accounts Home:

getList() - This function is used to get list of the accounts.

getCustomerList() - This function is used to get list of the customers.

editAccount() – This function is used to open the dialog box of update account.

getAccountName() – This function is used to get the name of a particular account.

deleteAccount() – This function is used to delete the account.

plotOnMap() – This function is used to open the map plotting component.

closeDialog() – This function is used to close the dialog box.

backButton() – This function is used to go back.

Add-Account:

getCustomerName() - This function is used to get the name of the customer.

addAccount() – This function is used to post or put the values of the account into the database.

closePopup() - This function is used to close the popup.

generateRandomInteger() – This function is used to generate random integers for the account ID.

openGoogleMap() – This function is used to open the dialog box of the google map.

Account Detail:

getAccounById() – This function is to get account details by the primary key this is email.

backButton() – This function is used to the previous page.

Google Map:

markerDragEnd() – This function is used to get the values of longitude and latitude on change of the marker.

getAddress() – This function is used to get the address from the map using longitude and latitude.

closeDialog() – This function is used to close the dialog box.

saveLocation() – This function is used to save the location in the database.

Map Plotting:

getAccountList() – This function is used to get a particular customer by ID.

Logs:

ngOnInit() – This function is used to get the logs from the database and sort them in the descending order of time.

backButton() – This function is used to go to the previous page.

3.3.4 API Component

REST API:

Controller class:

CustomerController - The CustomerController is a component that is responsible for handling customer-related operations. It contains methods: fetchCustomerProfile(params), updateCustomersProfile(params), deleteCustomerProfile(params), postCustomerProfile(params). The fetchCustomerProfile(params) method is used to retrieve the profile of a specific customer based on the parameters passed to it. The updateCustomersProfile(params) method is used to update a customer's profile based on the unique customer's identifier passed to it. The deleteCustomerProfile(params) is used to delete the customer based on the id passed to it. The postCustomerProfile (params) is used to post the details of the new customer to the database.

AccountController – The Account controller is a component that is responsible for handling account-related operations. It contains methods: fetchAccountDetails (params), postAccountDetails (params), putAccountDetails (params), deleteAccountDetails (params). The fetchAccountDetails is used to get the details of a particular account. The postAccountDetail is used to post the details of the new account inside of a customer. The putAccountDetails is used to update any details related to a particular account. The deleteAccountDetail is used to delete the details of a particular account detail.

AdminController – This Controller is responsible for the authentication of the admin. It contains methods such as:

Get: This get request get the details of any registered user based on the email.

Register: This post request is responsible for the registration of the new user.

Authenticate: This post request is responsible to authenticate an already registered user.

Reset: This put request is responsible for the updating the password related to any user.

LogsController- The logs controller is a component that is responsible for handling logging-related operations. All the operations performed are logged using this component only. This component has 2 methods: getLogs which is responsible to view all the logs, postLogs which is responsible to post any event occurred.

3.3.5 API Catalogue and Endppoints

Name of API	**Description**	**Access Level**
CDM_API	This API will be used to retrieve data from the database.	Only the Authorized users have access for retrieving the customer information and the information about the account.

Table 2 – API

Endpoints –

- https://localhost:4200/GET/api/Customers
- https://localhost:4200/GET/api/Customers/{gstin}
- https://localhost:4200/PUT/api/Customers/{gstin}
- https://localhost:4200/POST/api/Customers
- https://localhost:4200/DELETE/api/Customers/{gstin}
- https://localhost:4200/GET/api/Accounts/{email}
- https://localhost:4200/PUT/api/Accounts/{email}
- https://localhost:4200/GET/api/Accounts/{email}
- https://localhost:4200/POST/api/Accounts
- https://localhost:4200/DELETE/api/Accounts/{email}

Services:

CustomerService- In customer service, the CRUD operations involve managing customer data such as their name, contact information, and sectors of business. Creating a new customer record involves collecting and storing this information in a database.

AccountService- These operations include creating a new account record, retrieving account information, updating account details such as the account's revenue or contact information, and deleting account records.

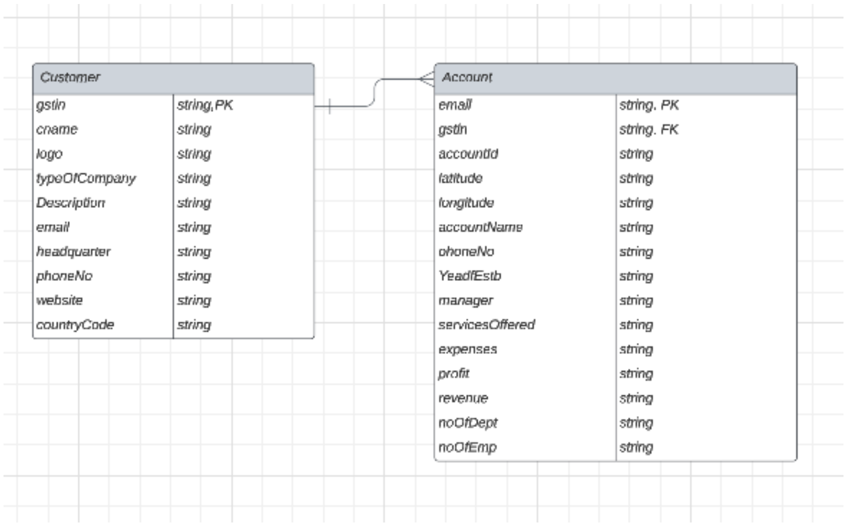
LogsService- This service is used to get and post the logs from the database.

SearchService- This service is used to implement the searchbar in the header component.

Auth Services: The Auth Services class provides authentication and authorization functionalities for users in a system. It typically includes methods for user login, logout, registration, and password management. The main methods are login and signup. The login method checks whether a user's credentials match the ones stored in the database and grants access to the user if they are authenticated. The signup method creates a new user account by collecting and storing their personal information and login credentials.

3.3.6 Data Design

3.3.6.1 List of Key Schemas/ Tables in Database



Customer	
gstin	string,PK
cname	string
logo	string
typeOfCompany	string
Description	string
email	string
headquarter	string
phoneNo	string
website	string
countryCode	string

Account	
email	string. PK
gstin	string. FK
accountId	string
latitude	string
longitude	string
accountName	string
phoneNo	string
YeadfEstb	string
manager	string
servicesOffered	string
expenses	string
profit	string
revenue	string
noOfDept	string
noOfEmp	string

Figure 3.5 – Data Design

3.3.6.2 Details of access levels on key tables in scope

The Admin and Accounts collections will not have access level restrictions applied based on how the admin is related to it.

- The Developers have admin access, and they are the ones, who are going to create the

account initially for a user, by storing the username and a default password which would be devised according to a simple logic which relates to the user's details.

- The Users (Fastenal registered users) will have access to create, view, update and delete customer records along with accounts. They will also have the access to view the location of the accounts of a particular customer in a map.

3.3.6.3 Key Design Consideration in Data Design

There are numerous important design factors to bear in mind while creating a data design model in MS SQL, including:

- Only one Customer can be created by using one GSTIN number, this ensures there are no duplicate customer in the database.
- No customer can have same accounts in the same location.
- No account can be repeated this is ensured using the email related to the account.

3.4 Exploring the Web Application

3.4.1 Authentication

Login Component

Login() – On clicking the Login button, this function will be called which will use the Email and Password entered by the user into the form, which will be retrieved using "form.value" to login(params) function in login Component typescript file.

As the user enters the webpage, the user will be shown a login screen. User needs to enter a valid username and password to access the dashboard page.

The login screen contains 2 text fields to enter email and password. There are various validations implemented in the login component like the email and password fields cannotbe empty and email id format validations are implemented. The login button won't work till both username and password fields are validated. Also, a toggle visibility button is implemented to show or hide password.

Register User Component

RegisterUser() – On clicking the Register button, this function will be called which will use the email, name, Password, number, entered by the user into the form, which will be retrieved using “form.value” to registerUser(params) in the registerComponent typescript file.

Once user clicks this, he is asked to fill the details in the form. There are double validation in these fields one is on the client side and other is on the server side. The password can be smaller than a fixed length and it has to be alphanumeric with a special character and phone number can only have 10 digits and email has to be unique. After these validations are passed then only a new user is registered. Also, a toggle visibility button is implemented to show or hide password.

Reset Password Component

ReserPasword() **-** On clicking the Reset password button, this function will be called which will use the email, newPassword, oldPassword entered by the user into the form, which will be retrieved using “form.value” to resetPassword (params) in the resetPassword typescript file.

3.4.1 Customer Dashboard

Dashboard is the page where the user lands after logging in successfully. There are optionsin the side pane to add a customer, view logs and sign out. The dashboard consists of list of customers and its details.

List of Customers

The main objective of the project is to display all accounts of a customer in a map.This page shows the list of the Customers with required details.

Angular Material has been used for designing the list of branches as the table. Angular Material is a User Interface (UI) component library that developers can use in their Angular projects to speed up the development of elegant and consistent user interfaces. Angular Material offers you reusable and beautiful UI components like Cards, Inputs, Data Tables, Date pickers, and much more.

Each component is ready to go with default styling that follows the Material Design Specification. Nonetheless, you can easily customize the look and feel of Angular Material components. The list of available Angular Material components continues to grow with each iteration of the library.

Searching

A text field is provided for searching the data using any keys. Searching has been implemented in a way that it will work for the page where we are in.

Account Dashboard

Dashboard is the page where the user lands after clicking a particular customer. There are optionsin the side pane to add an account, view logs and sign out. The dashboard consists of list of accounts and its details and plot on map option.

List of Accounts

The main objective of the project is to display all accounts of a customer in a map.This page shows the list of the Accounts with required details.
Angular Material has been used for designing the list of branches as the table. Angular Material is a User Interface (UI) component library that developers can use in their Angular projects to speed up the development of elegant and consistent user interfaces. Angular Material offers you reusable and beautiful UI components like Cards, Inputs, Data Tables, Date pickers, and much more.
Each component is ready to go with default styling that follows the Material Design Specification. Nonetheless, you can easily customize the look and feel of Angular Material components. The list of available Angular Material components continues to grow with each iteration of the library.

searching

A text field is provided for searching the data using any keys. Searching has been implemented in a way that it will work for the page where we are in.

Angular Google Maps Component

It is used for plotting the locations in the map with some information about the accounts of a customer.

This new component was released by angular, which is a new package that wraps up the Google Maps JS API. This again makes life easier and as now we don't need to code in JS and can use TypeScript for the purpose.

This module can be installed by using the command – `npm install @angular/google-maps`.

GoogleMapsModule gives us three components which can be used by us viz.

(a) GoogleMap: this is a wrapper for google maps, which can be accessed via the google-map selector.
(b) MapMarker: it is used to insert markers in the map, using the map-marker selector.
(c) MapInfoWindow: it is used to show an info window for a marker and can be accessedusing the map-info-window selector.

To add the maps JavaScript API, we need to use a script tag in the index.html file of the angular project. The API key needs to be generated from the Google Cloud Platform to use it in our application.

To achieve what the website has achieved when the map button is clicked, we need to create a separate angular component – say "accounts-in-map-component" and edit the html file to add the following.

First by adding the google map selector in the html file of the component, we add the template of the map to the UI. We can set the input properties like the height and width according to how we want to show the map in the UI. There are several methods defined for the google maps selector and can be used according to the need.

Next step is to add the map-marker to the template of the map, which can be done using the map-marker selector. There is a property named position in the map-marker selector which accepts an array of property of the form {lat: x, lng: y} and the markers will be positioned to those points.

The Label property of the map-marker can be used to describe the place near the map-marker. The title property of the map-marker can be set, and the value is visible on hover over a map marker.

We can also set an animation to the map-marker say bounce, which gives a pleasant view to the map.

The values of latitude and longitude are brought directly from the database, and they are iterated in the typescript file which helps in plotting the markers.

Finally, map-info-window selector allows you to display an info pane, while the marker is triggered for an event pre-defined, let us say mapClick. This is also demonstrated in the application, and we can see the branch code and the city name in the map-info-window.

Auth Guard

User is navigated to various pages with routes which uses URL's. There is a chance that user can enter the page URL where the user is not allowed to visit. To overcome this situation, we have a component called Auth Guard. Auth Guard is the function where it will be executed before the URL is visited. So that we can check whether the user is allowed to visit this page or not. If not, we can redirect the user to some other page showing that unauthorized access.

There is an argument called "canActivate" in routing module where we need to mention which class to be executed before visiting the URL. For child routes also there is a similar function where we can give whether to give the page access to user or not.

In our project we cannot allow user to access the dashboard and change password pages without the authentication. For this scenario we have implemented the Auth Guard. It contains two functions for routes and its children. We are checking if the user has logged in or not in Auth Guard function so that we can decide whether to allow user to access the page or not. We have created a page for Unauthorized access error.

3.4.2API Component

Application Programming Interface, or API, is a software bridge that enables communication between two programmes. You utilise an API every time you use a mobile app like Facebook, send an instant message, or check the weather.

We have created an API for the interaction between the front-end system and MSSQL. There are also some other functions like encryption and decryption in API. While sending the data when calling the API, it will be encrypted at client side and decrypted at server. HS256 has been used for this encryption and decryption. If any attacker got the data in the server, it will be in encrypted format and it guarantees the security while sending data over API.

There are 3 API routes we have implemented based on the requirement. There is a controller class contains the routes of every API calls. Controller class is the first class where it hits when an API has been called.

Validate credentials route: This route is designated for validating credentials while logging in. Username and password will be passed through JSON body in an encrypted format. It needs to be decrypted first and then validate the credentials with the credentials we retrieve from database. If the credentials are correct, then the user document will be returned in response to API call.

Encryption and decryption

HS256 Encryption

HMACSHA256 is a type of keyed hash algorithm that is constructed from the SHA-256 hash function and used as a Hash-based Message Authentication Code (HMAC). The HMAC process mixes a secret key with the message data, hashes the result with the hash function, mixes that hash value with the secret key again, and then applies the hash function a second time. The output hash is 256 bits in length.

An HMAC can be used to determine whether a message sent over an insecure channel has been tampered with, provided that the sender and receiver share a secret key. The sender computes the hash value for the original data and sends both the original data and hash value as a single message. The receiver recalculates the hash value on the received message and checks that the computed HMAC matches the transmitted HMAC.

Any change to the data or the hash value results in a mismatch, because knowledge of the secret key is required to change the message and reproduce the correct hash value. Therefore, if the original and computed hash values match, the message is authenticated.

HMACSHA256 accepts keys of any size, and produces a hash sequence 256 bits in length.

3.5 Architecture Diagram

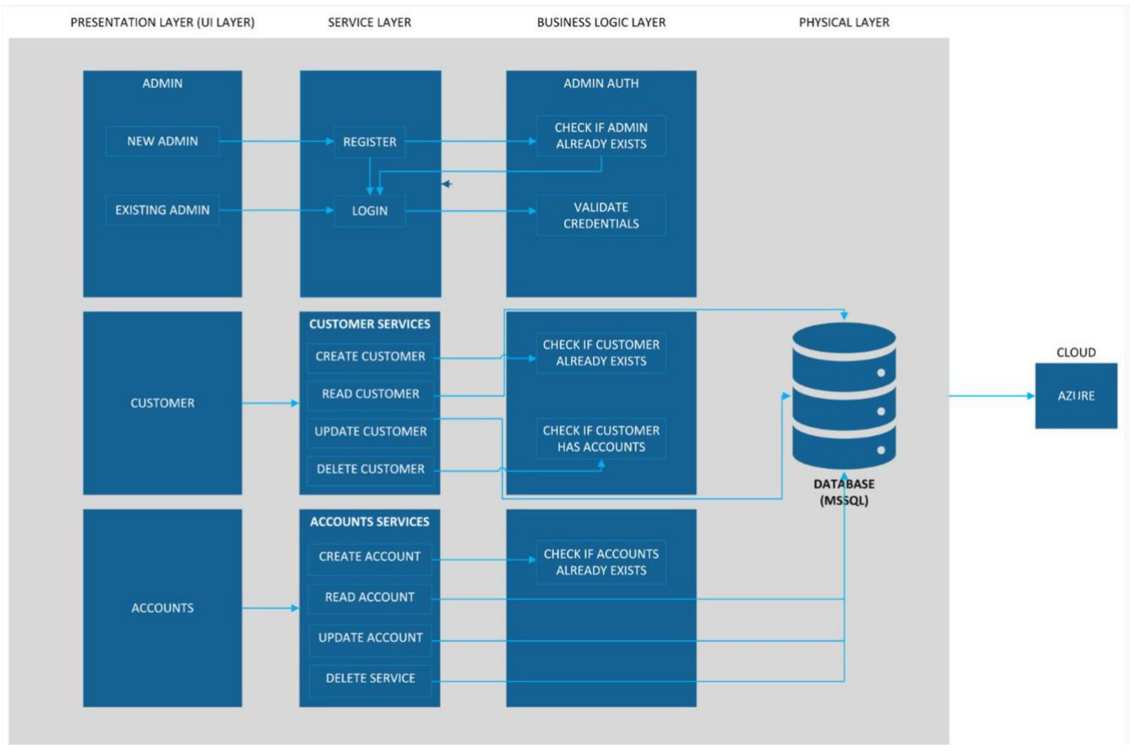


Figure 3.6 – Architecture Diagram

3.6 Conclusion

This chapter has properly explained the high-level design documentation, low-level design documentation, and has completely given an overview on various parts of the application. This will help anyone who reads the report and give them an idea on how to replicate the application or improvise it.

Chapter 4

Result Analysis

4.1 Introduction

This chapter is written to analyse the completed web application. Screenshots of the web app has been included and each screenshot has its description and will help the readers to understand the design of the application. It will also give an insight on what can be improved in the same.

Finally, a section on significance of the results obtained is also included, which talks about the user interface of the web app, and how the design helps the user in knowing about his/her branch in their home state.

4.2 Result Analysis – Web Application

4.2.1Register Page

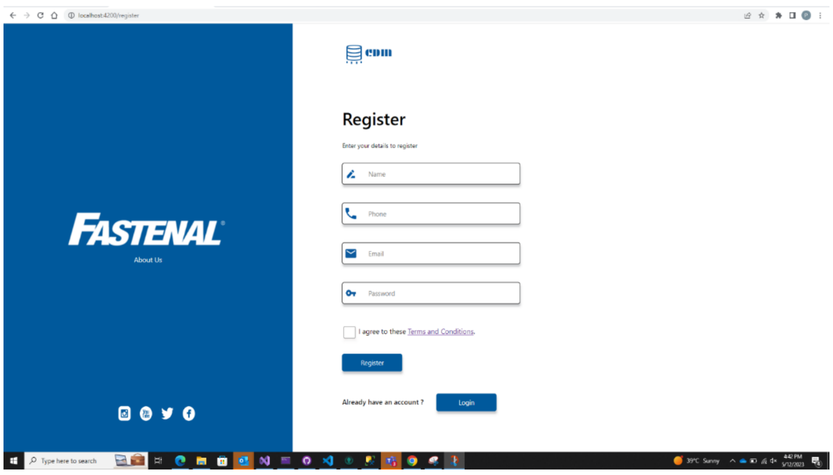

cdm
Register
Enter your details to register
Name
Phone
Email
Password
I agree to these Terms and Conditions.
Register
Already have an account ?
Login
FASTENAL
About Us

Figure 4.1 – Register Screen

This is the register page for any new user, here user can register itself to gain access to the site. This page contains multiple fields and all these fields are validated on both client level and server level.

4.2.2 Log IN Page

This is the first page which a user will see once he opens the portal. As the user enters the website, the user will be greeted with the login screen. After logging into the website, the customer dashboard will open.

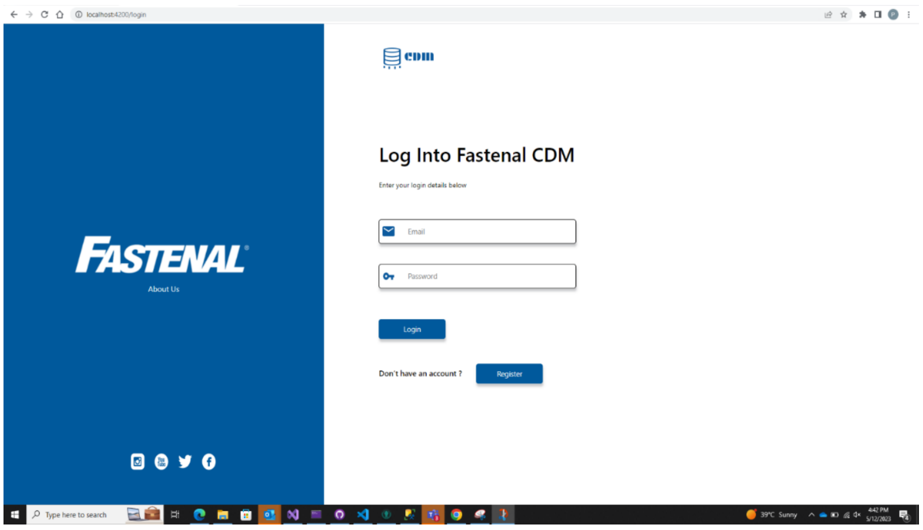


Figure 4.2 – Login Screen

4.2.3 Customer Dashboard

This is the first page where the user will be directed after the user has successfully logged in. This page contains all the information about the customers of the company. This page has multiple operations related to customers like update, add and delete existing customers.

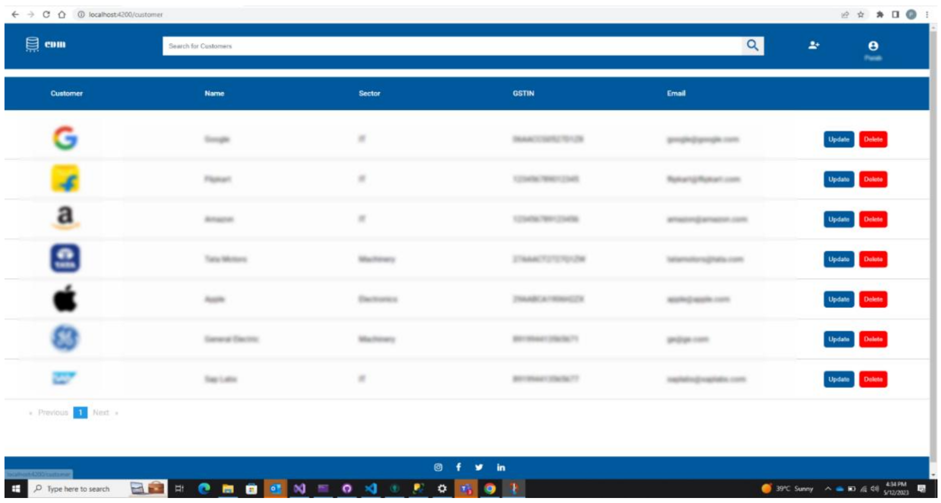


Figure 4.3 - Customer Dashboard

4.2.4 Add Customer Form

This is add customer page. Here we can add new customers. It contains real time validations on all the required fields, so that no irrelevant information related to customer can be added to the database, which can cause problems at the later stages.

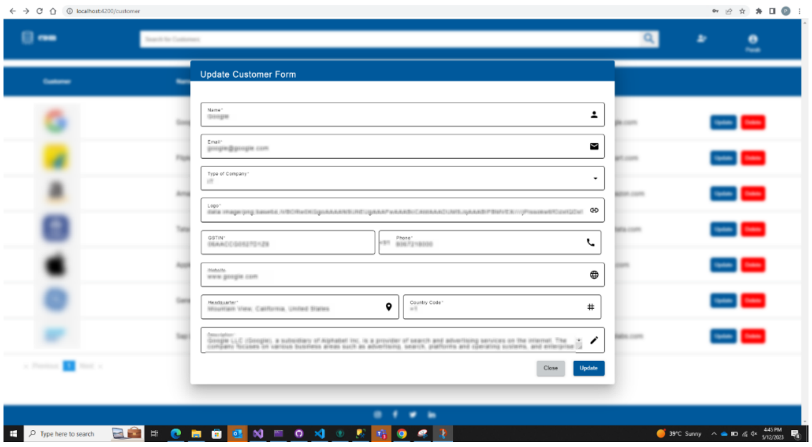


Figure 4.4 – Add Account Form

4.2.5 Update Customer

This is the update customer page. Here we can update the details of the existing customer.

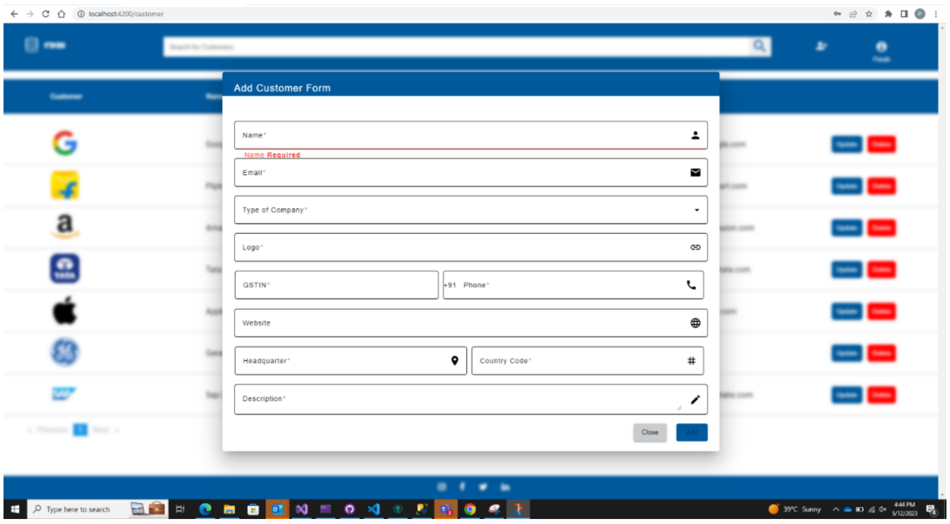
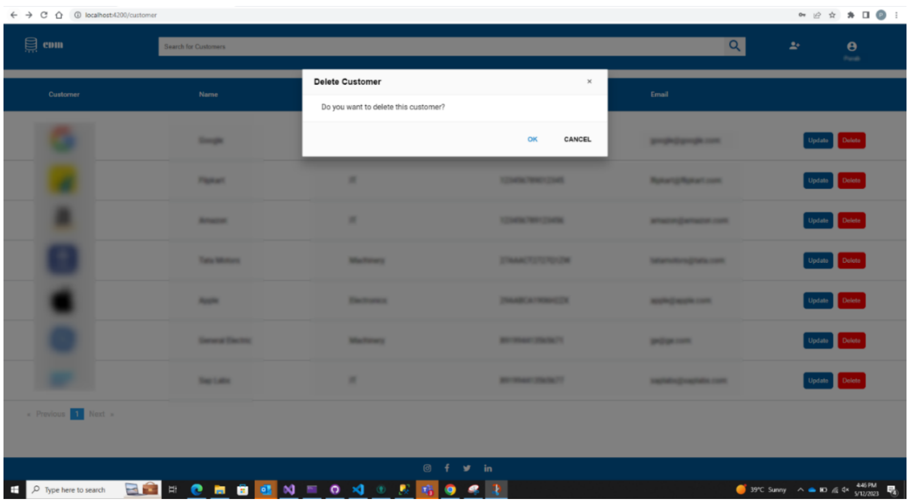


Figure 4.5- Update Customer Form

4.2.6 Delete Customer

This is the delete customer function. Here we get the functionality to delete the customer if it as no accounts linked to it. Is there are accounts present in that customer then admin is not allowed to directly delete the accounts.



Delete Customer

Do you want to delete this customer?

Figure 4.6 – Delete Customer

4.2.7 Account Home

This is account home page. Once admin clicks on a particular customer, he will be redirected to this page. Here some additional information about the customer is displayed and all the accounts linked to that customer is showed.

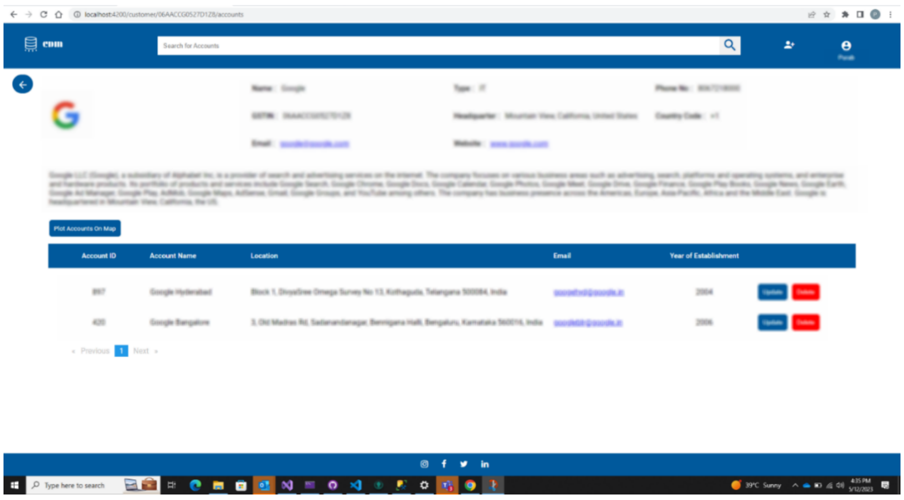


Figure 4.7 – Account Home

4.2.8 Add Account

This is the add account pop up. Using this admin can add accounts to that customer. All the fields in this are validated according to the requirement and for the location of the account, google API is used. Using this admin can point the exact location of the accounts.

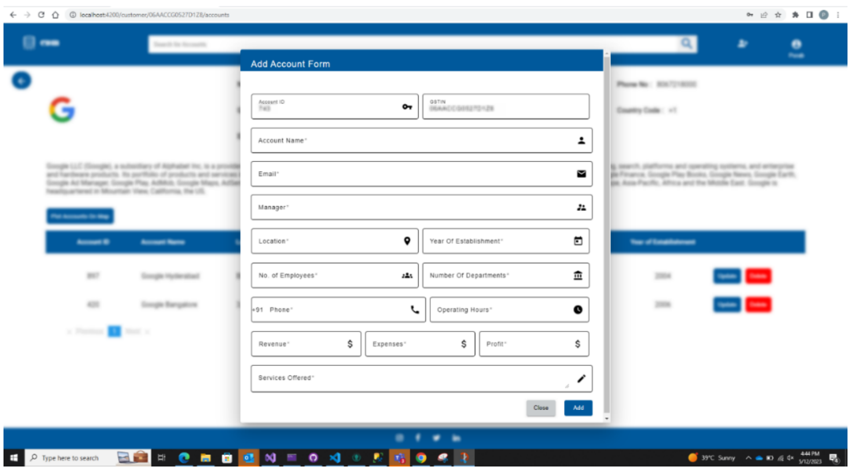


Figure 4.8 – Add Account

4.2.9 Update Account

This is the update account pop up. Using this admin can update the details of the existing account.

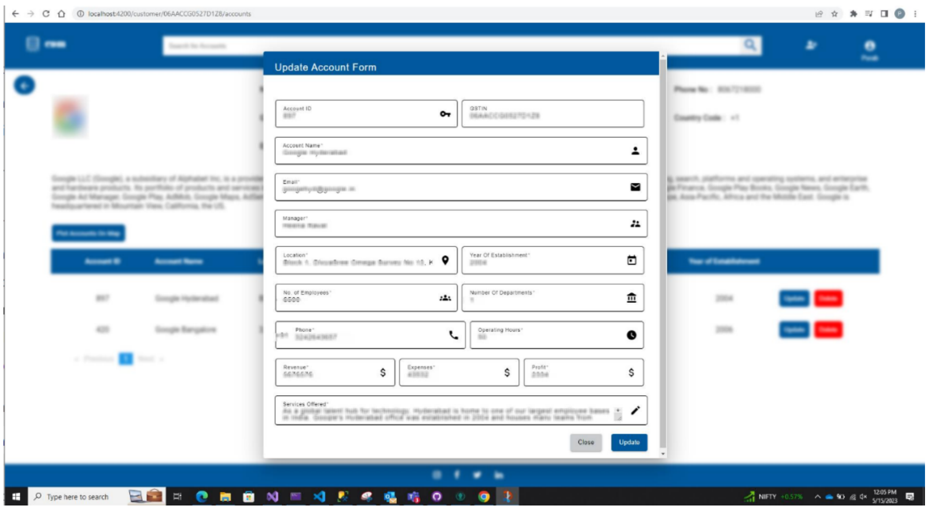


Figure 4.9 -Update Account

4.2.10 Accounts Plot

At the account home there is a button provided to plot all the accounts of that particular customer on the map. By clicking on that button another popup will open, that will have all the accounts plotted on the map. This will be implemented using google map API for angular.

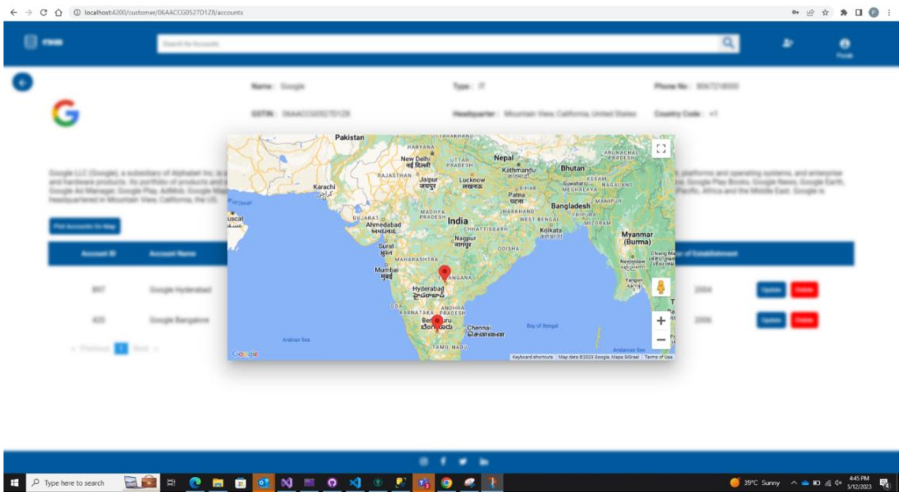
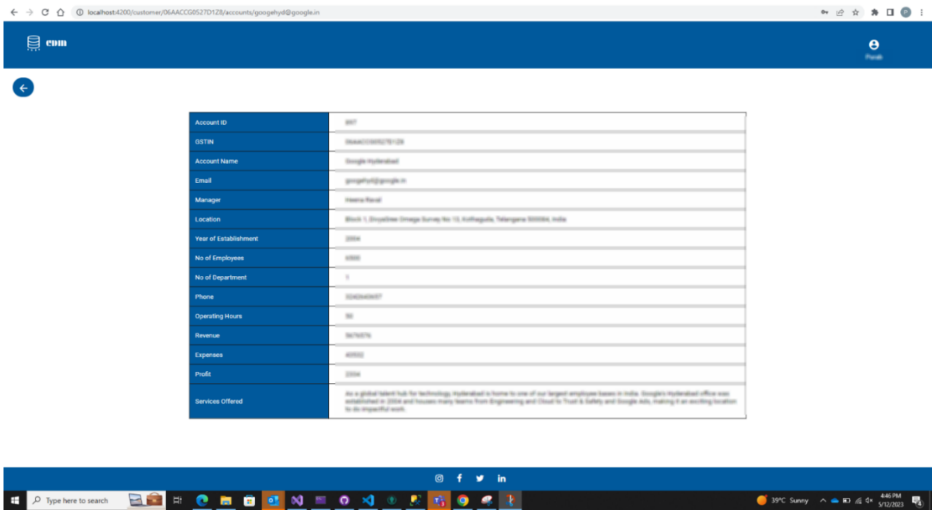


Figure 4.10 – Account Plot

4.2.11 Account details

Once admin clicks on a particular account, he will be redirected to account dashboard. This account dashboard will display all the details related to that account which was not displayed earlier in the account home page.



localhost:4200/customer/06AACCG0527D1Z8/accounts/googehyd@google.in

Account ID	[illegible]
GSTIN	[illegible]
Account Name	Google Hyderabad
Email	googlehyd@google.in
Manager	[illegible]
Location	[illegible]
Year of Establishment	2004
No of Employees	[illegible]
No of Department	1
Phone	[illegible]
Operating Hours	[illegible]
Revenue	[illegible]
Expenses	[illegible]
Profit	[illegible]
Services Offered	[illegible]

Figure 4.11 – Account Details

4.2.12 Navbar

In the navbar there are some other functionalities such as search bar implementation, admin options like logs, logout and reset password.

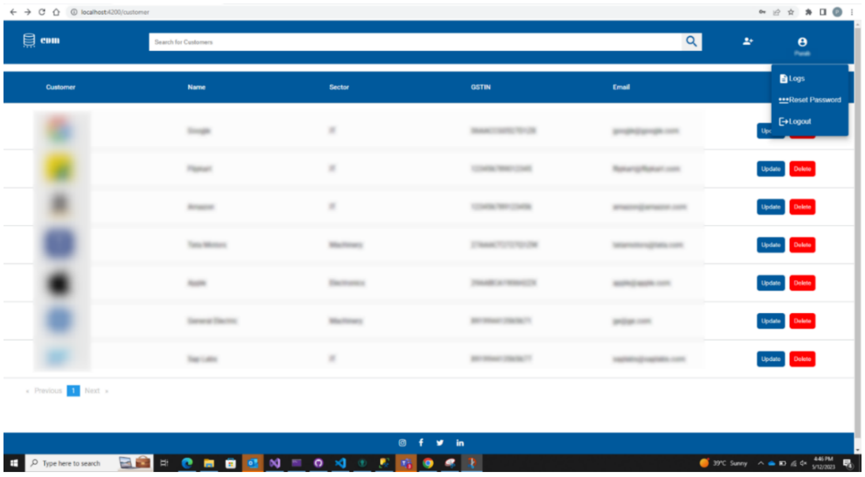
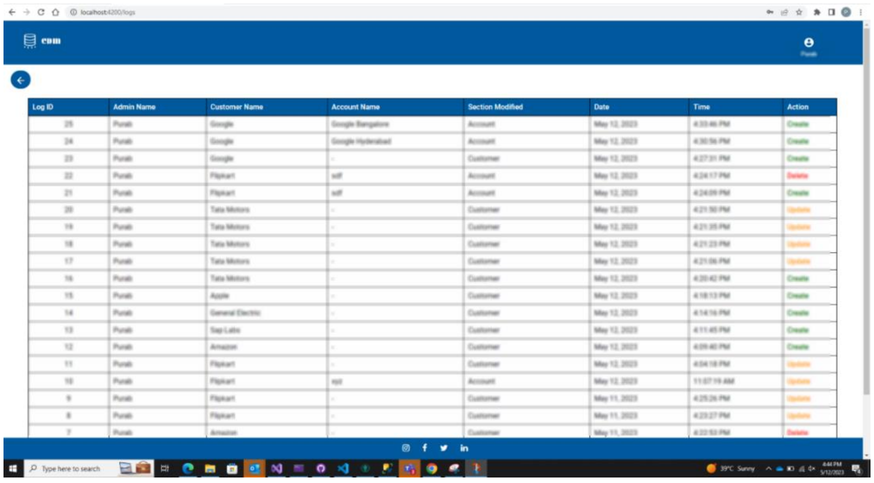


Figure 4.12 - Navbar

4.2.13 Logs

This is functionality provided in the navbar of all the pages in the website. Using this admin can see all the operations performed by all the admins till that time. No admin can edit the logs, this is a read-only functionality.



Log ID	Admin Name	Customer Name	Account Name	Section Modified	Date	Time	Action
25	Purab	Google	Google Bangalore	Account	May 12, 2023	4:33:46 PM	Create
24	Purab	Google	Google Hyderabad	Account	May 12, 2023	4:30:56 PM	Create
23	Purab	Google	-	Customer	May 12, 2023	4:27:31 PM	Create
22	Purab	Flipkart	sdf	Account	May 12, 2023	4:24:17 PM	Delete
21	Purab	Flipkart	sdf	Account	May 12, 2023	4:24:09 PM	Create
20	Purab	Tata Motors	-	Customer	May 12, 2023	4:21:50 PM	Update
19	Purab	Tata Motors	-	Customer	May 12, 2023	4:21:35 PM	Update
18	Purab	Tata Motors	-	Customer	May 12, 2023	4:21:23 PM	Update
17	Purab	Tata Motors	-	Customer	May 12, 2023	4:21:06 PM	Update
16	Purab	Tata Motors	-	Customer	May 12, 2023	4:20:42 PM	Create
15	Purab	Apple	-	Customer	May 12, 2023	4:18:13 PM	Create
14	Purab	General Electric	-	Customer	May 12, 2023	4:14:16 PM	Create
13	Purab	Sap Labs	-	Customer	May 12, 2023	4:11:45 PM	Create
12	Purab	Amazon	-	Customer	May 12, 2023	4:09:40 PM	Create
11	Purab	Flipkart	-	Customer	May 12, 2023	4:04:18 PM	Update
10	Purab	Flipkart	xyz	Account	May 12, 2023	11:07:19 AM	Update
9	Purab	Flipkart	-	Customer	May 11, 2023	4:25:26 PM	Update
8	Purab	Flipkart	-	Customer	May 11, 2023	4:23:27 PM	Update
7	Purab	Amazon	-	Customer	May 11, 2023	4:22:53 PM	Delete

Figure 4.13 -Logs

4.2.14 Reset password

Reset-Password is another functionality provided in the navbar. Using this admin can reset his login credentials. For this he need to enter the old password and new password and new password must also pass all the validations that are set for the password, only after this the password will be updated and after that admin will be logged out of the account.

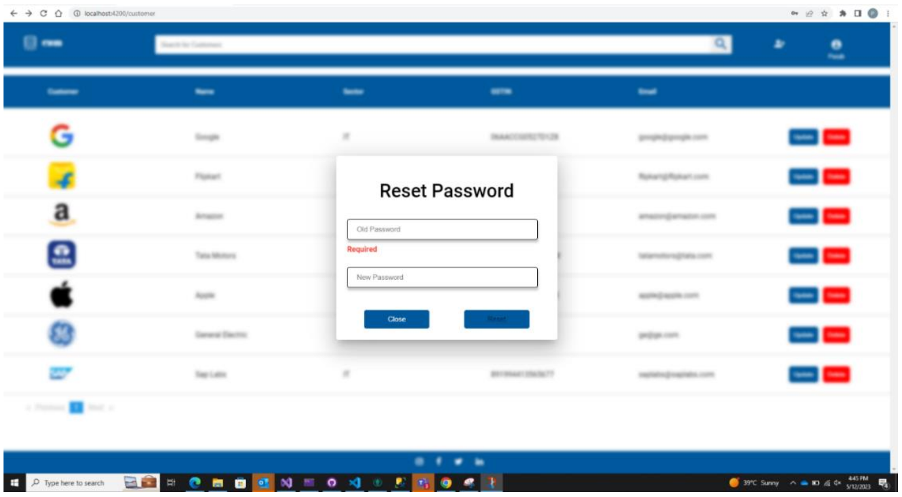
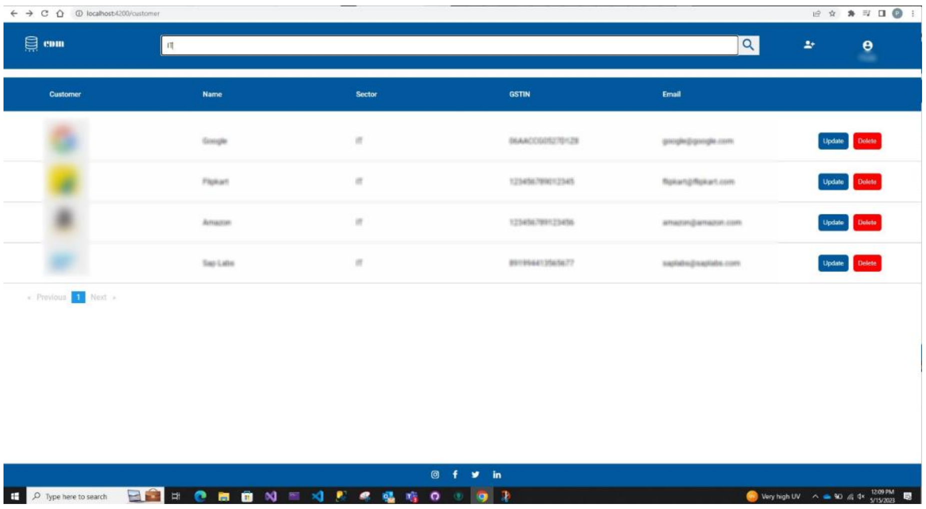


Figure 4.14 – Reset password

4.2.15 Search bar Implementation

This is another functionality in the navbar that is search bar implementation. Admin can search on all the fields that are there on that page and the records satisfying the search keyword will be displayed.



Customer	Name	Sector	GSTIN	Email	
	Google	IT	06AACCG0527D1Z8	google@google.com	Update Delete
	Flipkart	IT	123456789012345	flipkart@flipkart.com	Update Delete
	Amazon	IT	123456789123456	amazon@amazon.com	Update Delete
	Sap Labs	IT	891994413565677	saplabs@saplabs.com	Update Delete

Figure 4.15 Search Bar

4.3 Significance of the results obtained

Every customer of Fastenal is associated with accounts. This website will help the customers to list the accounts of Fastenal.

It also allows the users of this website to look the locations of the accounts in a map, where the markers when clicked greets the user with branch code and city.

User authentication is also successfully implemented which acts as a security to the web app and necessary steps have been taken to validate the change password credentials in the client side itself, which makes the website more secure.

Auth Guards are also implemented, which doesn't allow unauthorized users to go to a page of the website like dashboard.

4.4 Conclusion

In conclusion, the results obtained through the web application offers:

- Seam-less experience for the user with proper login system.
- Use of client-side validation to check the data at runtime.
- Single page web application using Angular to provide faster interface.
- Use of MSSQL to store the data.
- Creating a custom REST (Representational state transfer) API to undertake various methods.
- Proper integration of frontend and backend via API.
- The application keeps track of the users already registered with the company and helping them view details.
- The application uses native angular google maps component to plot the locations of the accounts.
- The reset password feature allows the user to change the password at a later point of time.
- Use of Angular Material Components.

This project aims at increasing the knowledge about various technologies and thinking about the ways in which user experience could be enhanced. It also helps in understanding various security aspects that are involved with a web application.

Chapter 5

Conclusion and Future Scope

5.1 Brief Summary and Conclusion Specific to Web App

The primary goal of this thesis is to include information about the layout and operation of a web application. The Fastenal: Customer Data Management web application, on which I worked during my internship is the combination of all the technologies that the company might have to deal with. The current web application is an extended version with a plenty of features and database schemas.

The application that I have worked on is a platform that offers a convenient experience to the user in browsing through the various branches that are there in their resident state. In actual scenario as well, the company works to provide a seamless experience to the customers while they go through the procedure.

This project is a subset of the Branch Sales (Point of Sales) POS application and is solely for learning purpose.

This project aims at increasing the knowledge about various technologies and thinking about the ways in which user experience could be enhanced. It also helps in understanding various security aspects that are involved with a web application.

5.2 Future Scope

5.2.1Design Changes

A few of the design changes can be implemented further to make the application look more robust and beautiful. In the future versions of the application, we will try to include a card view with the image of the branch, it is address and other details which a user view upon clicking on any branch in the list view of the branches.

5.2.2Server-Side Pagination

Another change that can be implemented is the server-side pagination with sorting and filtering features.

5.3.1 Maps with Navigation

Another feature that can be implemented to the application to make it more accessible is the feature to add navigation to the map view feature of the accounts. When a user clicks on any marker of the account on the map, an option to provide navigation directions from the user's current location to that account can be provided.

References

[1] T. Williams, “Ultimate ASP.NET Core Web API Development Guide,” O’Reilly Online Learning, https://www.oreilly.com/library/view/ultimate-aspnet-core/10000DIVC2022122/ (accessed Feb 13, 2023).

[2] M. Schwarzmüller, “Understanding typescript – 2020 edition,” O’Reilly Online Learning, https://learning.oreilly.com/videos/understanding-typescript/9781789951905/ (accessed Feb 26, 2023).

[3] M. Schwarzmuller, “Angular - the complete guide [2021 edition],” O’Reilly Online Learning, https://learning.oreilly.com/videos/angular-the/9781788998437/9781788998437-video18_8/ (accessed March 6, 2023).

[4] I. Griffiths, “Programming C# 8.0,” O’Reilly Online Learning, https://learning.oreilly.com/library/view/programming-c-8-0/9781492056805/ch14.html (accessed April 18, 2023).

[5] Wadepickett, “Tutorial: Create a web API with ASP.NET CORE,” Microsoft Learn, https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-7.0&tabs=visual-studio (accessed April 21, 2023).

[6] Anonanon, gbngbn 69.5k88 gold badges161161 silver badges241241 bronze badges, MarianMarian 15.3k22 gold badges5656 silver badges7373 bronze badges, and James PulleyJames Pulley 34322 silver badges44 bronze badges, “What are some best practices for using schemas in SQL Server?,” Database Administrators Stack Exchange, https://dba.stackexchange.com/questions/4075/what-are-some-best-practices-for-using-schemas-in-sql-server (accessed April 25, 2023).

[7] “Angular google maps (AGM),” Angular Google Maps Components, https://angular-maps.com/ (accessed May 6, 2023).

[8] Angular, https://angular.io/ (accessed May 6, 2023).