

Fastenal - Branch Location Manager

*A Practice School Report submitted to
Manipal Academy of Higher Education
in partial fulfilment of the requirement for the award of the degree of*

BACHELOR OF TECHNOLOGY

in

Computer Science & Engineering

Submitted by

Ghanashyam R K P

180905042

Under the guidance of

Vamsi Krishna Reddy Pallamala
IT Associate Manager
Fastenal India

Dinesh Acharya
Professor, CSE
MIT Manipal



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

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

June 2022



MANIPAL INSTITUTE OF TECHNOLOGY

MANIPAL

(A constituent unit of MAHE, Manipal)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Manipal
15/06/2022

CERTIFICATE

This is to certify that the project titled **Fastenal – Branch Location Manager** is a record of the Bonafide work done by **Ghanashyam R K P** (*Reg. No. 180905042*) submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology (B.Tech.) in **COMPUTER SCIENCE & ENGINEERING** of Manipal Institute of Technology, Manipal, Karnataka, (A Constituent Institute of Manipal Academy of Higher Education), during the academic year 2021-2022.

Prof. Dinesh Acharya

*Professor, CSE Dept.
M.I.T, MANIPAL*

Prof. Dr. Ashalatha Nayak

*HOD, CSE Dept.
M.I.T, MANIPAL*

Project / Internship Offer Letter

Project Completion Letter

ACKNOWLEDGMENTS

First, I would like to thank my external guide, **Vamsi Krishna Reddy Pallamala – IT Associate Manager, Fastenal, India**, for providing appropriate suggestions on my work at company from time to time and encouraging me to achieve more.

Next, I would extend my thanks to **Akhila Janardhana Shastry – Team Lead, Branch Sales, Fastenal India**, for guiding me throughout the internship with trainings required, and providing appropriate feedbacks, which helped me to work with vigour till the end.

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

At last, I would like to thank **Dr. Ashalatha Nayak - HOD, CSE, Dr. Renuka Prabhu**, Practice School Coordinator and my Internal guide **Dinesh Acharya**, Prof. CSE, for supporting me throughout the internship with college related communication and assessments.

ABSTRACT

The project – **Fastenal: Branch Location Manager**, is a proof-of-concept project, assigned by Branch Sales – Team of Fastenal India. This project deals with listing of branches (stores), in the home state of a customer of Fastenal and plots all the locations in a map highlighting the branch code and the city of the branch location. This proof-of-concept project is aimed to test our technical skills from the company on Angular, ASP.NET 5, Angular Material and MongoDB.

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 branches 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, we were able to show a beautiful animation of markers and display branch code and city information on clicking the markers. 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 5 - Web API for backend, and MongoDB as the database. We also used Postman and Swagger UI for API-Testing purposes.

		Page No
Acknowledgement		i
Abstract		ii
List Of Tables		iv
List Of Figures		v
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 and/or LITERATURE REVIEW		4
2.1	Introduction to Project Title	4
2.2	Exploring the Tech Stack	5
2.3	Exploring the Software Tools	7
2.4	Conclusion	9
Chapter 3 METHODOLOGY		10
3.1	Introduction	10
3.2	High – Level Design	10
3.3	Low – Level Design	13
3.4	Exploring the Web Application	19
3.5	Conclusion	26
Chapter 4 RESULT ANALYSIS		27
4.1	Introduction	27
4.2	Result Analysis – Web Application	27
4.3	Significance of Results Obtained	33
4.4	Conclusion	33
Chapter 5 CONCLUSION AND FUTURE SCOPE		35
5.1	Brief Summary of Work	35
5.2	Future Scope	35
REFERENCES		36
PROJECT DETAILS		37
PLAGIARISM REPORT		38

LIST OF TABLES

Table No	Table Title	Page No
1.1	Project Schedule	2
3.1	API Catalogue	17

LIST OF FIGURES

Figure No	Figure Title	Page No
3.1	Process Flow Diagram	11
3.2	Sequence Diagram	12
3.3	Angular Class Diagram	15
3.4	API Class Diagram	16
3.5	Database Schema	18
3.6	AES Encryption	35
4.1	Login Screen	26
4.2	List View of Branches	27
4.3	Search Functionality	28
4.4	Branch Details	28
4.5	Map View of Branches	29
4.6	Branch Details in Map	30
4.7	Change Password Validation	30
4.8	Change Password Screen	31
4.9	Password Validation	31
4.10	Unauthorized Screen	33

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 centres 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 develop an application giving the ability to users of the app to view the branches of Fastenal within their home state. One of the interesting projects that the team is currently working on includes Customer Master Database Management (CMDM) and Branch Location Manager - which is in the planning phase.

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: Branch Location Manager” 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 explained minimally.

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 being planned for development, and it would help the customers of a particular branch to have a handy information of what other branches exist in the same state of their home branch. So, it'll be easy for customers to know details about all the branches of their state, which would ease their procurement of products from the store and make navigation easier.

The objective of this project is to allow customers of a particular branch to be able to authenticate themselves by logging in into the app and be able to get the information of all the branches present in their home state. They would be able to sort and filter that information. The details of the branches have its status of operating, so it'll help the customers in figuring out which stores are open, and which are not. Finally, by plotting the locations in a map, we give the customer an overview of where and how close are the branches in holistic view and, the customer would be able to click on a location and know it's Branch Code and City.

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 March 28th. It's marked as the 13th Week after my internship started at Fastenal

Topics	
Week 13	Discussion of Project Objectives
	Documentation of High-Level Design Document
Week 14	Review and approval of High-Level Design Document
	Discussion for Low Level Design Documentation
Week 15 and 16	Creation and Completion of Low-Level Design Document with multiple reviews following it.

Week 17 and 18 (Mid)	<p>Starting of the API and creation of database of the application</p> <p>Completing the API with required action methods.</p> <p>Completed final review of the API.</p>
Week 18 (Mid) and 19	<p>Started creating Front-End UI components</p> <p>Completion of all the Front-End UI Components and integration of both API and front-end.</p>
Week 20	Final Review and presentation of the project to supervisors of our team.

Table 1.1 – 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

Branch Location Manager is a web application that provides an interface to the user for logging in and viewing branch details of their state.

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 Dashboard which consists of two tabs to toggle between branches table – which contains the branches in the home state and map's view – which has all the locations plotted in the google map, with a map-view displaying the branch code and the city.

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 branch 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 MongoDB 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 programmes 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, colours, 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 programmes are also valid TypeScript programmes.

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 programmes, 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 MongoDB

MongoDB is a document-oriented database that is open source and cross-platform. MongoDB is a NoSQL database that stores data in JSON-like documents with optional schemas. MongoDB is a database that was created by MongoDB Inc. and released under the Server-Side Public License (SSPL).

All major programming languages and development environments have official MongoDB drivers. Other programming languages and frameworks have a plethora of unofficial or community-supported drivers.

The mongo shell has been the database's principal interface. MongoDB Compass has been available as a native GUI since MongoDB 3.2. Products and third-party projects provide administrative and data viewing user interfaces.

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 used as the frontend in the MEAN stack, which contains the MongoDB database, Express.js web application server framework, Angular (or AngularJS), and the Node.js server runtime environment.

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 the 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.

Visual Studio supports 36 programming languages, and the code editor and debugger can handle practically any programming language if a language-specific service is available (to varying degrees). C, C++, C++/CLI, Visual Basic.NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML, and CSS are among the built-in languages. Plug-ins are available for other languages such as Python, Ruby, Node.js, and M. Previously, Java (and J#) were supported.

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, categorise, 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, GraphQL, and RAML standards. 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 MongoDB Atlas

MongoDB Atlas is a MongoDB multi-cloud data platform. A well-maintained cloud database for modern applications is at its core. MongoDB, the most popular non-relational database, is best handled using Atlas. Because documents map directly to the objects in your code, MongoDB's document model is the simplest way to innovate. As a result, working with them is easier and more natural. You can store data in any structure you want, and you can alter your schema as your apps grow.

Atlas Database is available in over 80 AWS, Google Cloud, and Azure locations. Multi-cloud and multi-region deployments are now feasible, giving you the flexibility to select the providers and locations that best fit your customers. Uptime, scalability, and adherence to the toughest data security and privacy standards are all provided through best-in-class automation and tried-and-true methodologies.

2.3.4 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 customisable 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.

2.3.5 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 Mongo DB.

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: Branch Location Manager 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 MongoDB to retrieve, insert, delete, and update the database. It will also be using the Angular Material Component Library.

3.2.2 Tools Used

- Angular 13.3.0: 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 5
- MongoDB 5.0.6: A NoSQL database program.
- GitHub: Version Control System

3.2.3 General Constraints

The Fastenal: Branch Location Manager is user friendly and as automated as possible. The users can see the login page only after login. After logging in, users can view the branches within their home state.

3.2.4 System Design

3.2.4.1 Main Design Features

The main design features include 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

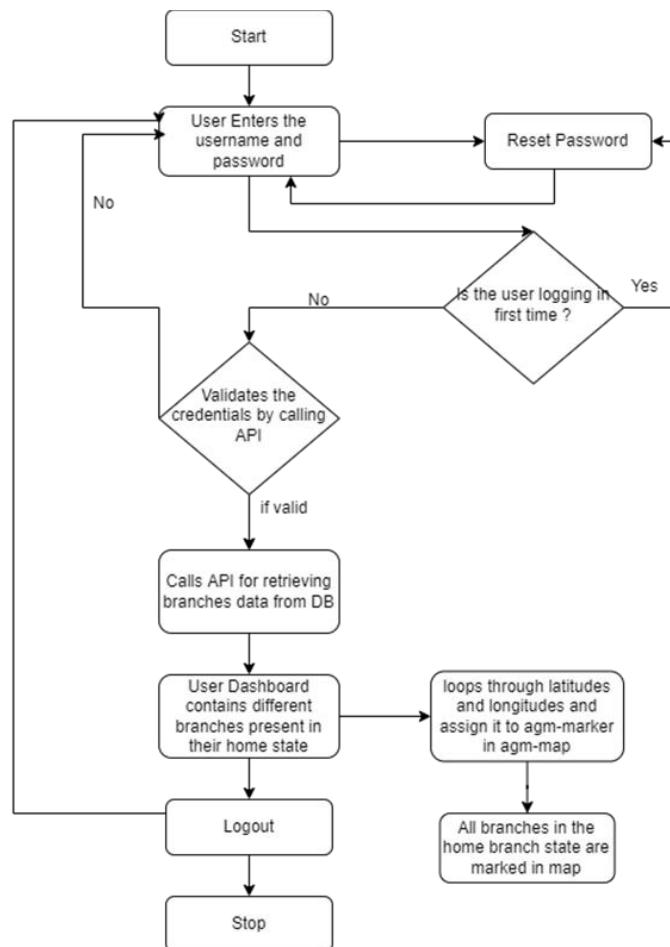


Figure 3.1 - Process Flow Diagram

3.2.4.3 Sequence Diagram

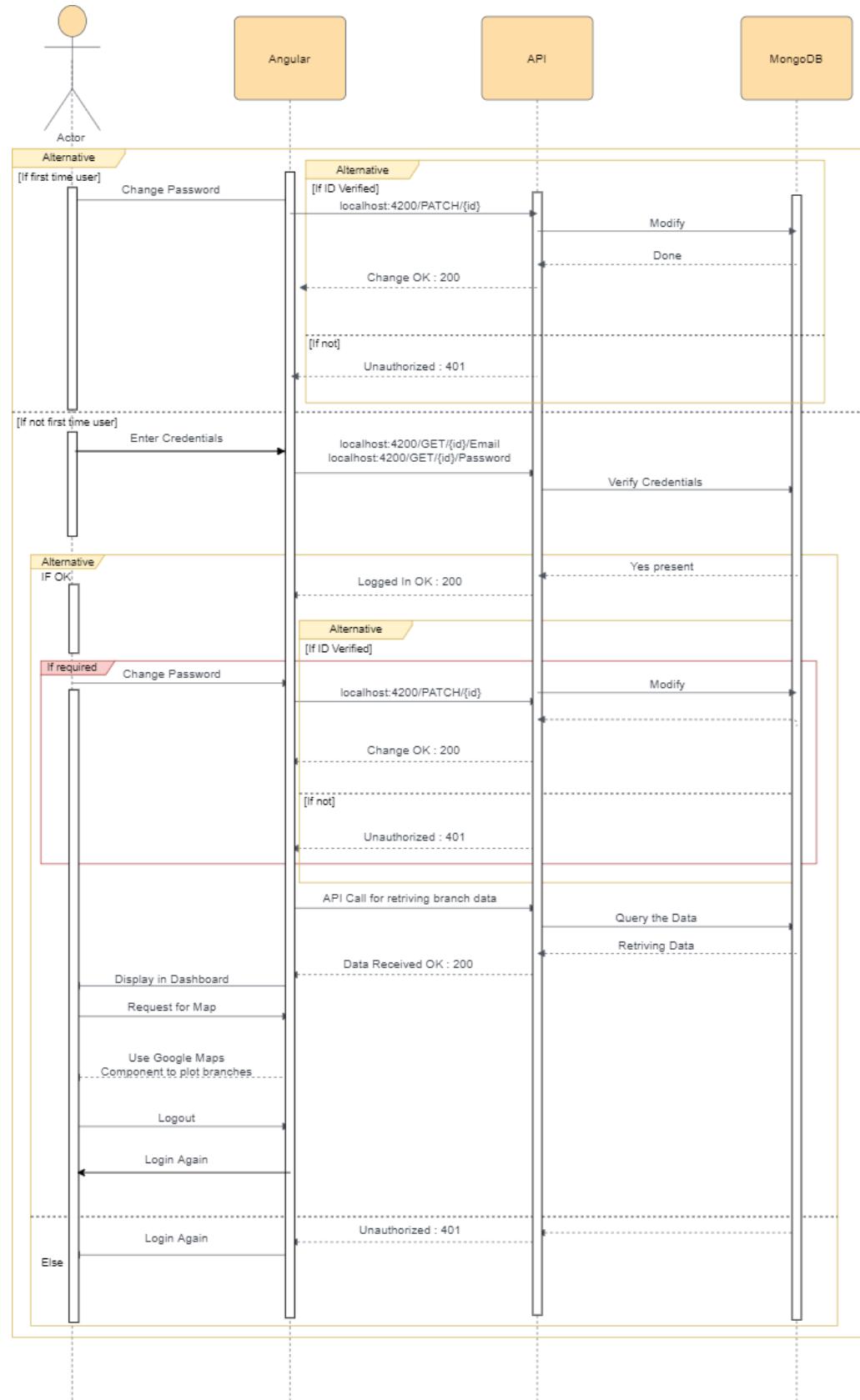


Figure 3.1 - Sequence Diagram

3.2.3 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 branches within their home state. 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 only the branches within his home state which is displayed on dashboard and can also filter and sort the data by every column mentioned in the list. We can access the branches located in maps from dashboard.

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.

3.2.4 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.

3.3 Low Level Design

3.3.1 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 branches within their home state. 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 only the branches within their home state, which is displayed on dashboard and can also filter and sort the data by every column mentioned in the list. The listed branches in the Data Table can also be viewed in the google map, for which we are using native angular maps component.

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.

3.3.2 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.

3.3.3 Components Design

3.3.3.1 Angular Component

User View Model: This is a view model class which is used for rendering the views. It acts as view model for User Data.

Branch View Model: This is a view model class which is used for rendering the views. It acts as view model for Branch Data.

Services: Any method we need to contact with API is written in services class. Any common methods can also be written in services class so that it can be available to all classes by injecting the service.

Login: Login Page takes username and password as inputs and validate credentials by calling API and logs the user into the dashboard if credentials are valid. Redirects to change password if the user is logging in first time.

Change Password: Every user is given an option to change their password. Inputs are the username, new password and old password input is taken depending on the first-time user or not.

Branches-In-Map: This is a component that contains a map locating all the locations of branches needed.

Dashboard: This is the page user lands when they login. It contains the branches in the state where the user is registered. This page will have two tabs in its body, One has good pagination view with all the branches contained in it and other page with map locating all the branches in the state where user registered.

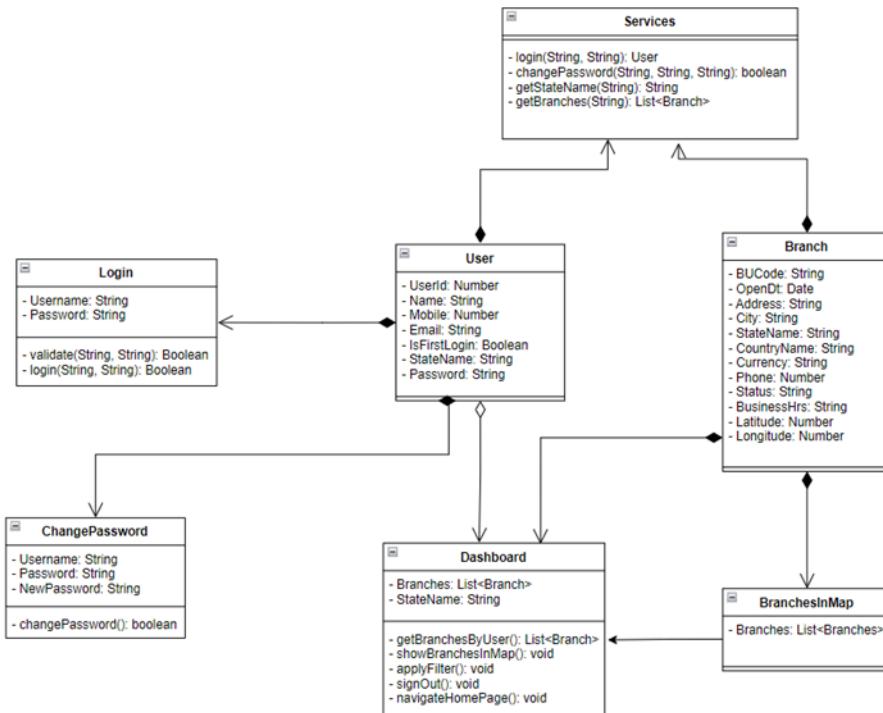


Figure 3.2 - Angular Class Diagram

3.3.3.2 API Component

User Model: This is a model class for the user collection in the database.

Branch Model: This is a model class for the branches collection in the database.

Services: This class will act as a helper class for the controller class, which will be having functions implemented, and they will be directly used in the controller class. This would make the class cleaner and since the functions are called many times, it becomes modular as well.

Controller: API Controller class is the class where all the needed HTTP methods would be defined. API hits these methods when an API is called with respective to their HTTP methods. We can also provide routes for every method.

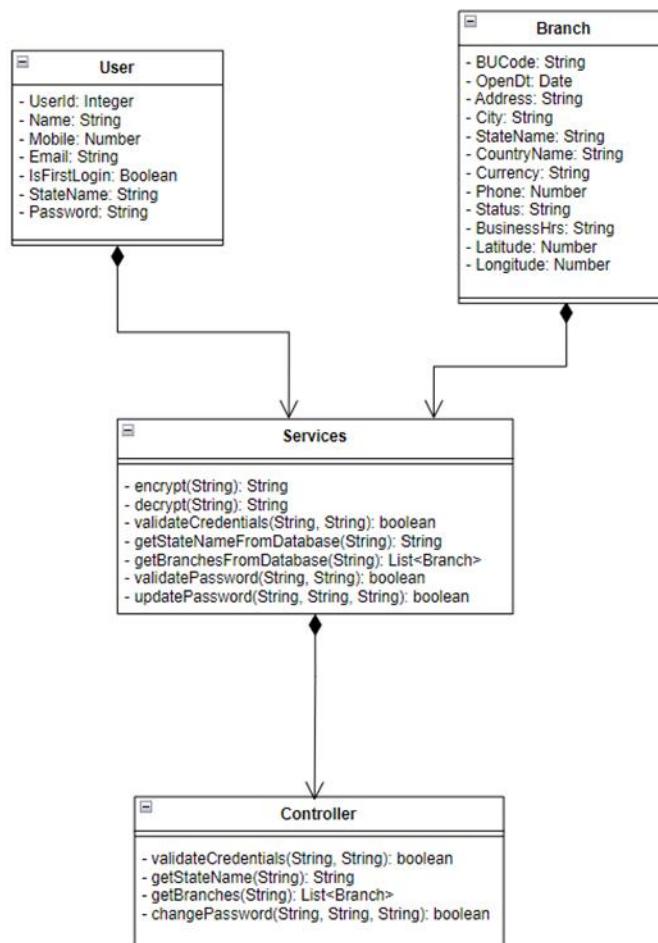


Figure 3.3 - API Class Diagram

3.3.3.3 Encryption Algorithm

For the encryption of credentials, we are using AES Encryption Algorithms.

It is based on ‘substitution–permutation network’. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations). Interestingly, AES performs all its computations on bytes

rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix.

Usage:

The credentials will be encrypted by this algorithm and will be stored in the database. While validation we'll retrieve the encrypted credentials and decrypt them and compare it to give access to the website.

3.3.3.4 API Catalogue and Endpoints

Name of API	Description	Access Level
branchUserAndAuth	This API will be used to authenticate, and retrieve data from the database	For Login purpose every user has access. Only the Authorized users have access for retrieving the branch's info and changing the password.

Table 3.1 – API Catalogue

Endpoints:

- <https://localhost:4200/POST/User/{Email}>
- <https://localhost:4200/PATCH/User/{Email}>
- <https://localhost:4200/GET/Branch>

3.3.4 Data Design

3.3.4.1 List of Key Schemas / Tables in the Database

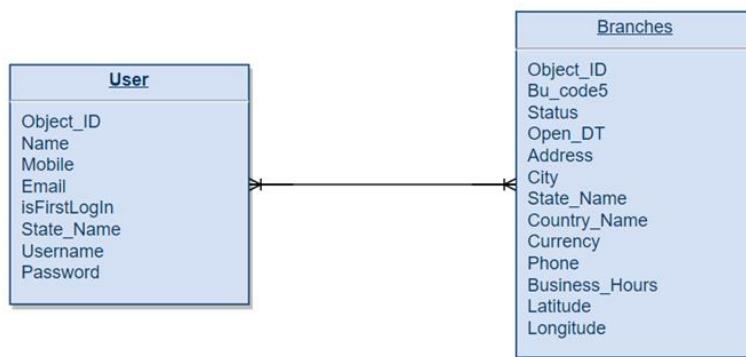


Figure 3.4 - Database Schema

3.3.4.2 Details of access levels on key tables in scope

The User and Branches collections will have access level restrictions applied based on how the user 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 view their data, and will only be able to edit the password, given that they enter a valid password that was in use before they want it to change.

3.3.4.3 Key Design Consideration in Data Design

In MongoDB, we can design a collection depending on how we use it while retrieval of data. Officially there are no rules that are documented by them, but they have given some recommendations.

There are two ways in which we can refer to Branch collection with the help of User collection.

- Embedding the respective branches that the user is associated with to the user collection in the form of an array
- Local reference – where we query the Branch collection for the branches which are associated with the respective users.

We have designed the collection in a way that supports the second approach of local reference. More details can be found in the system overview.

3.3.4.4 System Overview Detailed

This app starts with the login screen and as explained in the client-side validation part, validation is done and the credentials are sent to the API, where the credentials from the database are retrieved, and both are compared and authenticated accordingly.

There will also be a button for changing the password once logged in, and this page will do client-side validation as explained and make a change to the database using Patch method of the API to modify the collection in the backend.

In all other cases, they are greeted with a dashboard page and by default, we retrieve the data from the database using GET method of API and render it in the form of data table with pagination.

There will be a toggle button on the top with which one can toggle the view of data table to a map view, where we will be using native angular google maps component to render a cluster of locations of the branches in the map.

Finally, one can log out using the log out button, after which they will be greeted with a login window.

3.4 Exploring the Web Application

3.4.1 Authentication

3.4.1.1 Login Component

Login() – On clicking the Login button, this function will be called which will use the Username 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. If the user is logging in for the first time, after entering the default password (fastenal123), he'll be redirected to the change password screen first.

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

3.4.1.2 Change Password Component

ChangePassword() – On clicking the Reset Password button, this function will be called which will use the oldPassword, newPassword and confirmPassword entered by the user into the form, which will be retrieved using “form.value” to changePassword(params) in the changePassword Component typescript file.

After successful first-time login, or once user selects change password option from the header menu, the user will be redirected to the change password page. This component consists of three text fields, old password, new password and confirm password.

There are various validations implemented in this component as well. Such as length of new password must be a minimum of 8 characters, the new password and confirm password must match, the old password entered must be the valid password and all the fields must contain some values. The oldPassword cannot be equal to the newPassword. If all the validations are satisfied, then only the change password button would be enabled or otherwise, an error message will be displayed.

A toggle visibility button is also provided in all the text fields to show/hide values entered in every field. Another option to revert to the dashboard page is also available in this component.

3.4.2 Dashboard

Dashboard is the page where the user lands after logging in successfully. There are two options in the header i.e., change password and logout. Change password will navigate to change password component. And by clicking on logout the user will be logged out and will be navigated to login page. There is label on the header which states the name of user registered Fastenal branch state name.

There is a toggle switch between List of branches and Branches in Maps. These both list and maps are two different components integrated here.

3.4.2.1 List of Branches

The main objective of the project is to display all branches of the state where user is registered. This page shows the list of the branches 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.

There is a component called Data Table in Angular Material. Angular Data Tables is a library for building complex HTML tables that uses jQuery's Data Tables plugin. It is configured to support TypeScript and optimized for Angular 2+.

Angular DataTables will come in handy when:

- You have a very large dataset coming in from one or more API endpoints
- You need customized data sorting/filtering
- You need to export and print data from a table

Angular DataTables features can be broadly grouped into two sets: basic and advanced. From there, Angular DataTables also supports several extensions.

For most real-world applications, providing the table a `DataSource` instance will be the best way to manage data. The `DataSource` is meant to serve as a place to encapsulate any sorting, filtering, pagination, and data retrieval logic specific to the application.

A `DataSource` is simply a class that has at a minimum the following methods: `connect` and `disconnect`. The `connect` method will be called by the table to provide an `Observable` that emits the data array that should be rendered. The table will call `disconnect` when the table is destroyed, which may be the right time to clean up any subscriptions that may have been registered in the `connect` method.

Sorting has been implemented for all the required columns. There will be a small arrow icon which switches from increasing to decreasing and vice versa. Implementation of sorting can be done for every single column independently.

3.4.2.2 Filtering

A text field is provided for searching the data using any keys. Filtering has been implemented in a way that it will work for the page where we are in. This is because of server-side pagination. More about server-side pagination will be on next paragraph. With only one text field we can search for every field instead of having different text fields for each column.

3.4.2.3 Pagination

There are two types of pagination i.e., client-side pagination and server-side pagination.

Client-side pagination: client-side pagination is implemented after getting the whole data from database or any other source. So, if we have a huge amount of data, it will create a huge network traffic to retrieve the data.

Server-side pagination: server-side pagination works in a different way as compared to client-side pagination. It is more of a back end paginator. While retrieving the data itself it will load the data in pages and goes on by user requirement. By this network traffic is decreased as the retrieving is distributed and called separately by the user need. And we don't need to retrieve the whole data if user found the required data.

Every time user clicks on next page API is called for retrieving the data. The time taking for data to retrieve the page is obviously less than retrieving the whole data. So, the application works faster. Testing of server-side pagination can be done by the API calls in the network window in inspect in the browser.

3.4.2.4 Angular Google Maps Component

It is used for plotting the locations in the map with some information about the branch code and city.

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 accessed using 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 “branches-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. It will have a watermark throughout the map template saying, “For development purposes only” and it will vanish once a licence is bought for that API key.

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.

3.4.2.5 Route Guards

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.

3.4.3 API Component

API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API.

We have created an API for the interaction between the front-end system and MongoDB. 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. Caeser cipher 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.

Change Password route: This route is used for changing password. It will call other methods in services class. After decryption, the old password will be validated, and the new password will be updated if it returns true. Password stored in database will be encrypted using AES.

Get Branches route: This route is for retrieving the branches in the state where the user belongs. As we already have the state name of the user, we can retrieve the branches data. This data will be assigned to data source in data table in angular material.

3.4.4 Encryption and decryption

Before storing into the database, the password gets encrypted. Two types of encryption algorithms are used in the project

3.4.4.1 Caesar cipher

The Caesar Cipher technique is one of the earliest and simplest method of encryption technique. It's simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter some fixed number of positions down the alphabet.

For example, with a shift of 1, A would be replaced by B, B would become C, and so on. The method is apparently named after Julius Caesar, who apparently used it to communicate with his officials.

Thus, to cipher a given text we need an integer value, known as shift which indicates the number of positions each letter of the text has been moved down.

3.4.4.2 AES encryption

AES is an iterative rather than Feistel cipher. It is based on ‘substitution–permutation network’. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).

Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix

The number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key. The schematic of AES structure is given in the following illustration

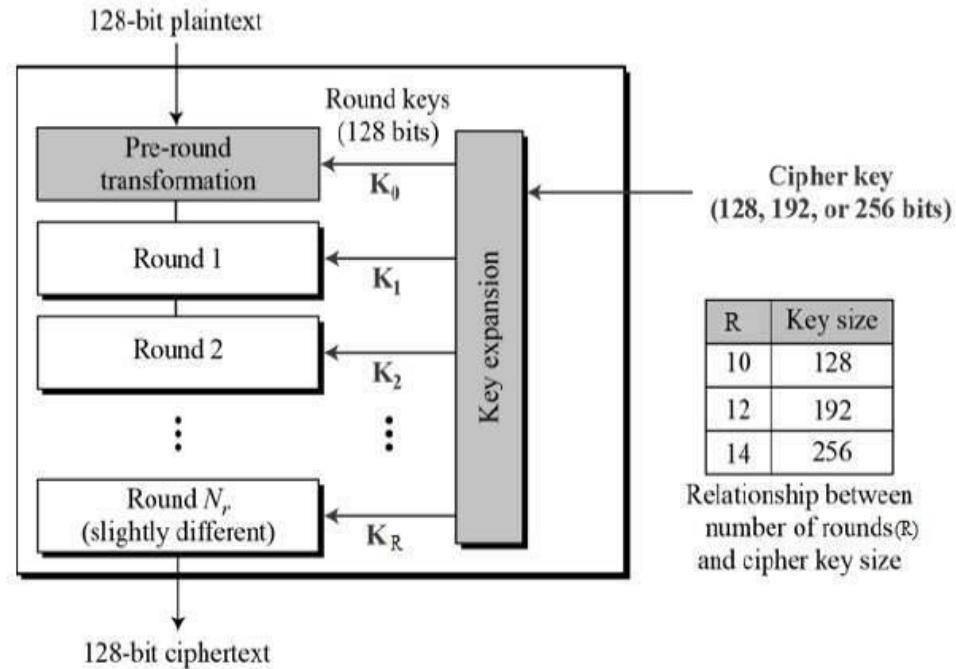


Figure 3.5 - AES Encryption

When the password is sent from angular frontend to the API layer, it gets encrypted using Caesar cipher, and after that when the entered password is sent from the API layer to the database, it again gets encrypted using AES encryption.

3.5 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

4.2.1 Login Screen

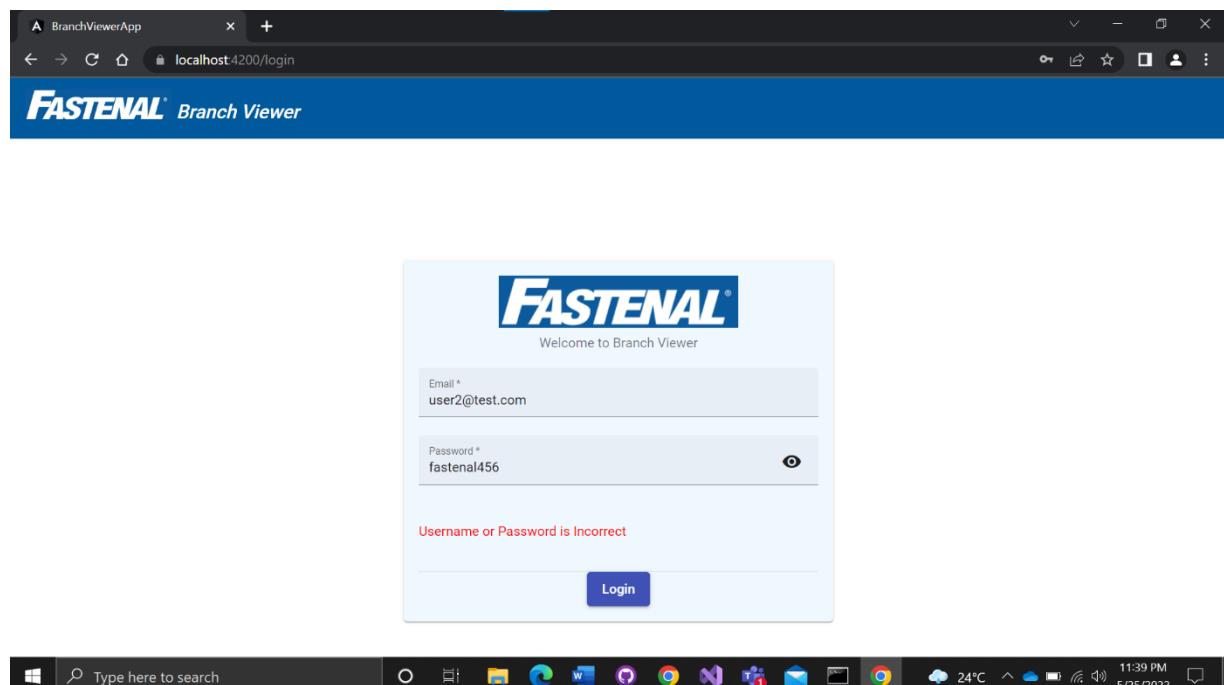


Figure 4.1 - Login Screen

As the user enters the website, the user will be greeted with the login screen. In this case since the username – user2@test.com does not exist, the component shows appropriate message for the user to correct it.

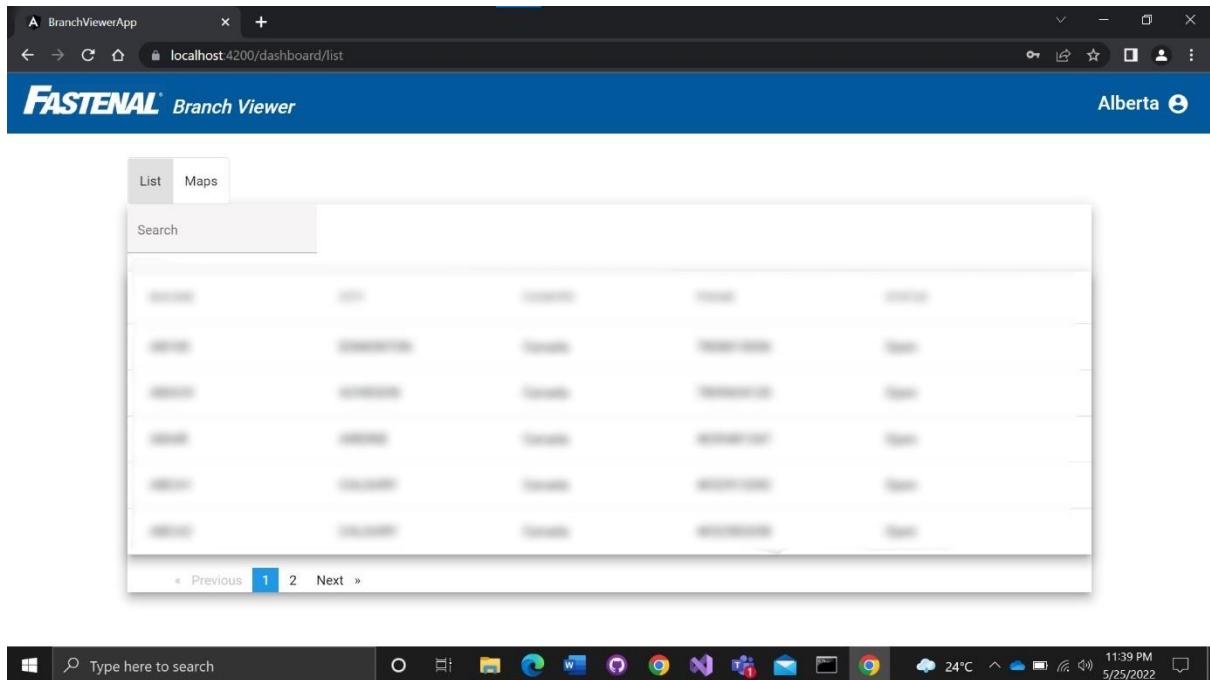


Figure 4.2 - List View of Branches

The above image shows the list view of the application, the data is blurred due to NDA. Angular Material DataTable is used for the purpose and server-side pagination implemented can be seen from the bottom of the data table.

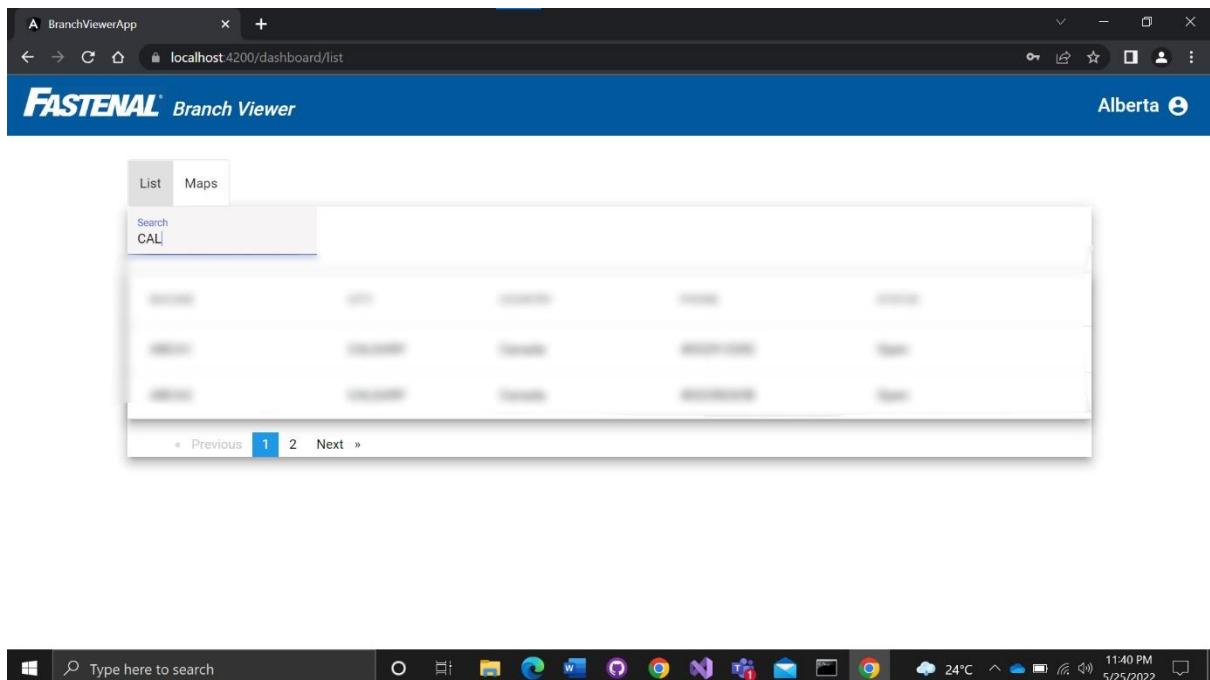


Figure 4.3 - Search Functionality

Search functionality implemented works well, for the page being returned by MongoDB. Before implementation of server-side pagination, client-side pagination was performed with the help of angular material and the search functionality was working for the complete data as well.

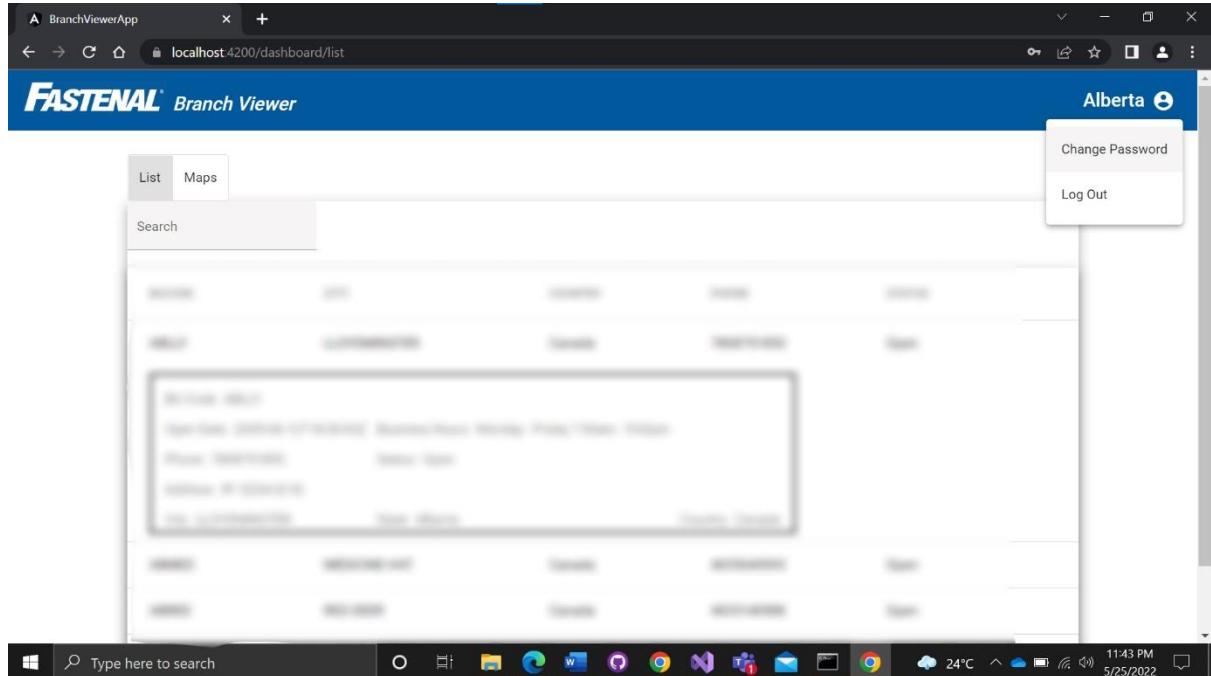


Figure 4.4 - Branch Details in a view pane

As a row of the table is clicked, we can see it expand and a details pane opens below the clicked row. This pane reflects upon the missed data about the branch, which is clicked. Again, the table is blurred due to NDA reason.

Also, we can see a dropdown menu for changing the password and logout functionality.

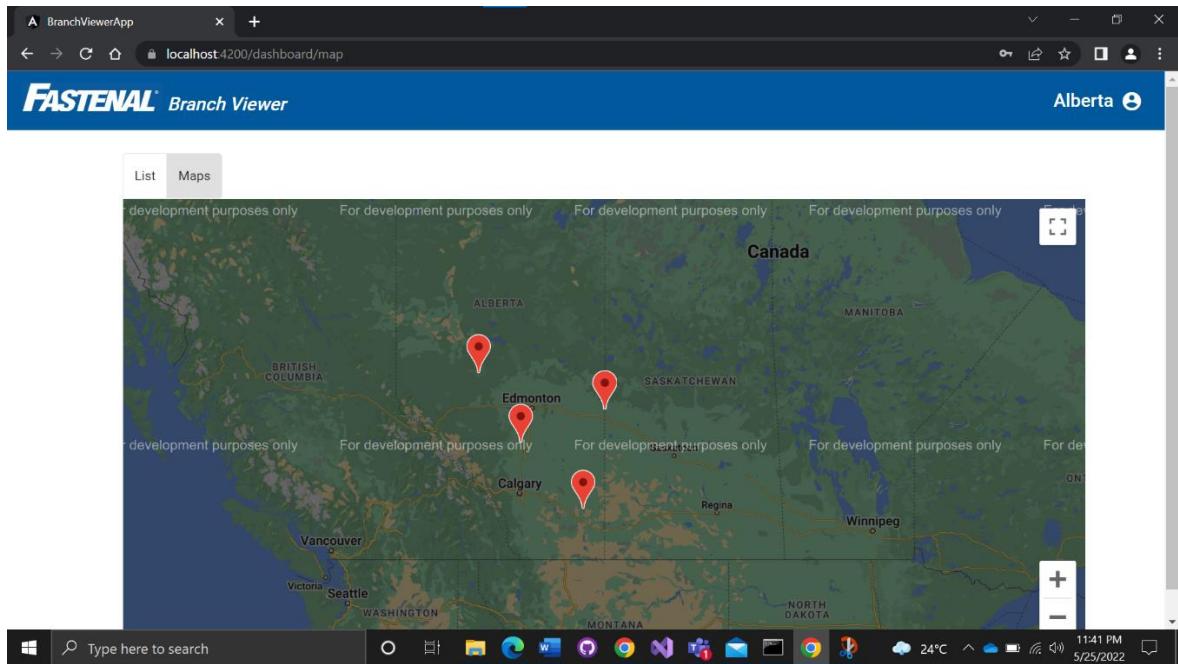


Figure 4.5 - Map View of Branches

The above is the map-view for the branches in the page of the page table. There is an animation of markers (bounce) implemented, which cannot be seen in the image.

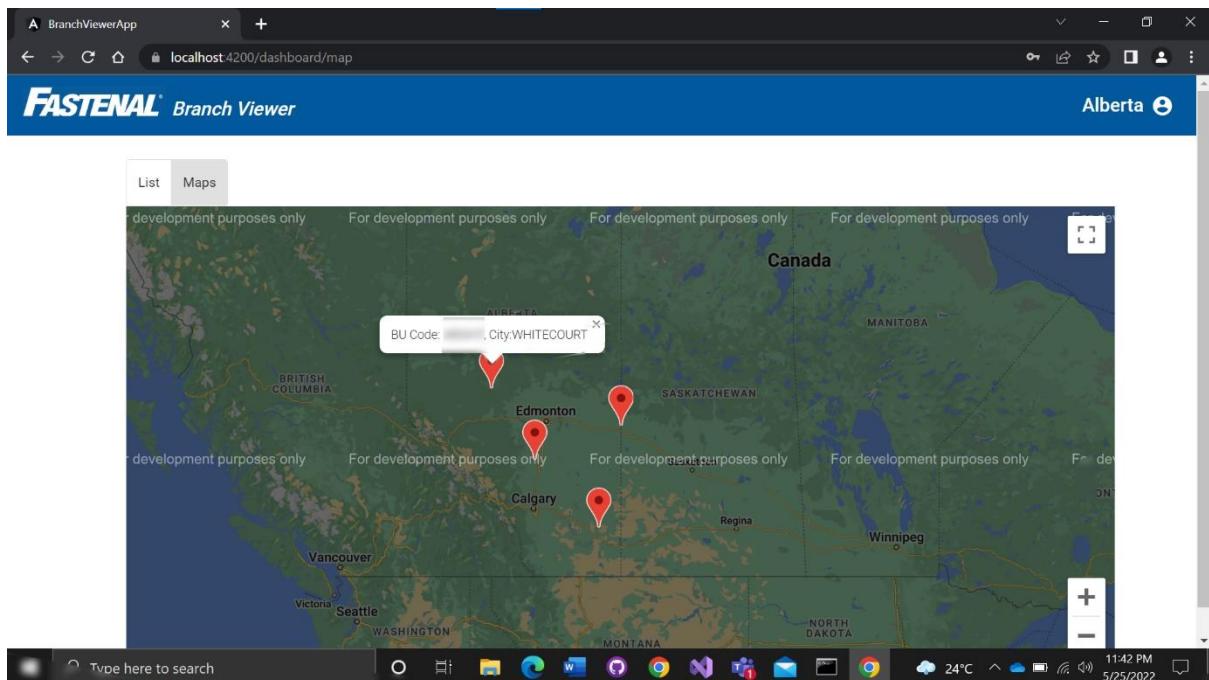


Figure 4.6 - Branch Details in Map

Upon clicking of the markers, the user is greeted with a map-info-window, which shows the branch code and the city of that marker

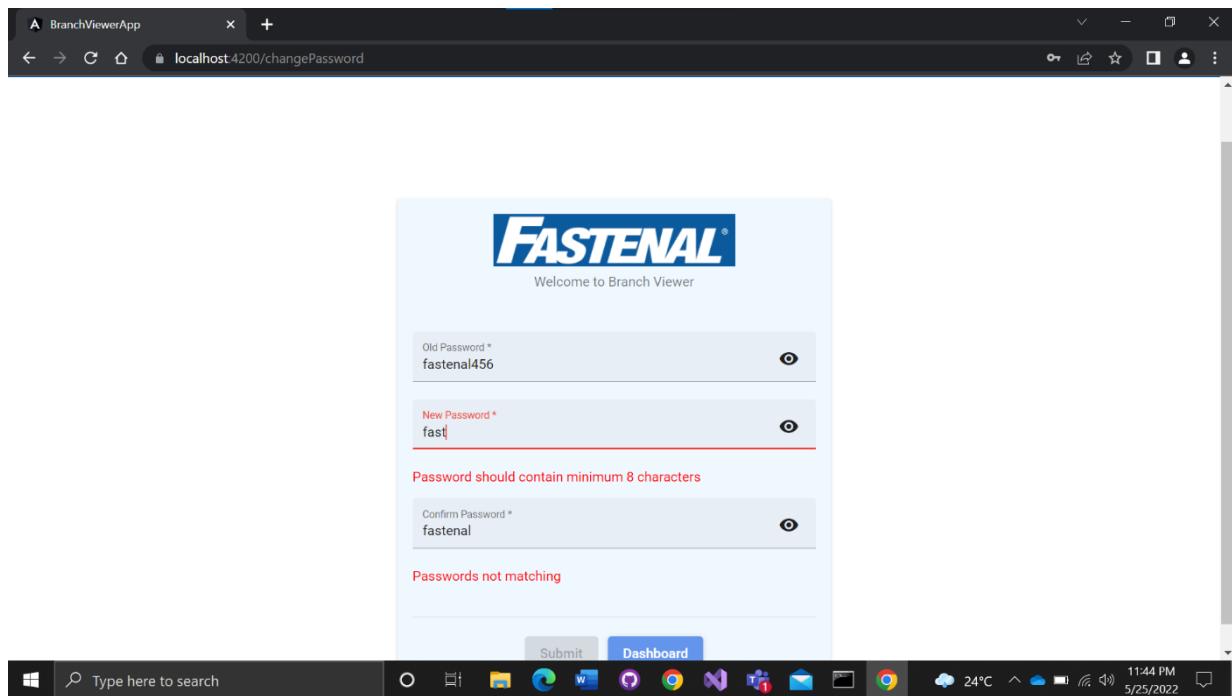


Figure 4.7 - Change Password Validation

Change password component with various validations like the new password should contain a minimum of 8 characters. Also, new password and confirm password should match, for the submit button to get enabled.

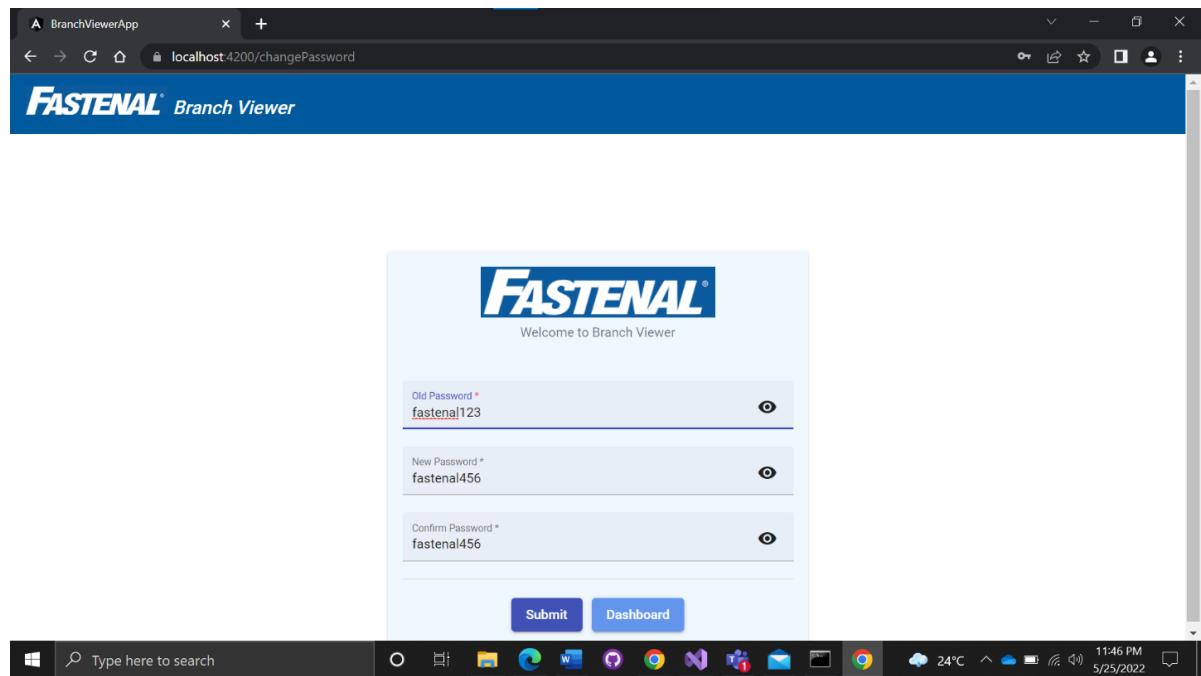


Figure 4.8 - Change Password Screen

Submit button will be enabled only if the value of new password and confirm password text fields matches

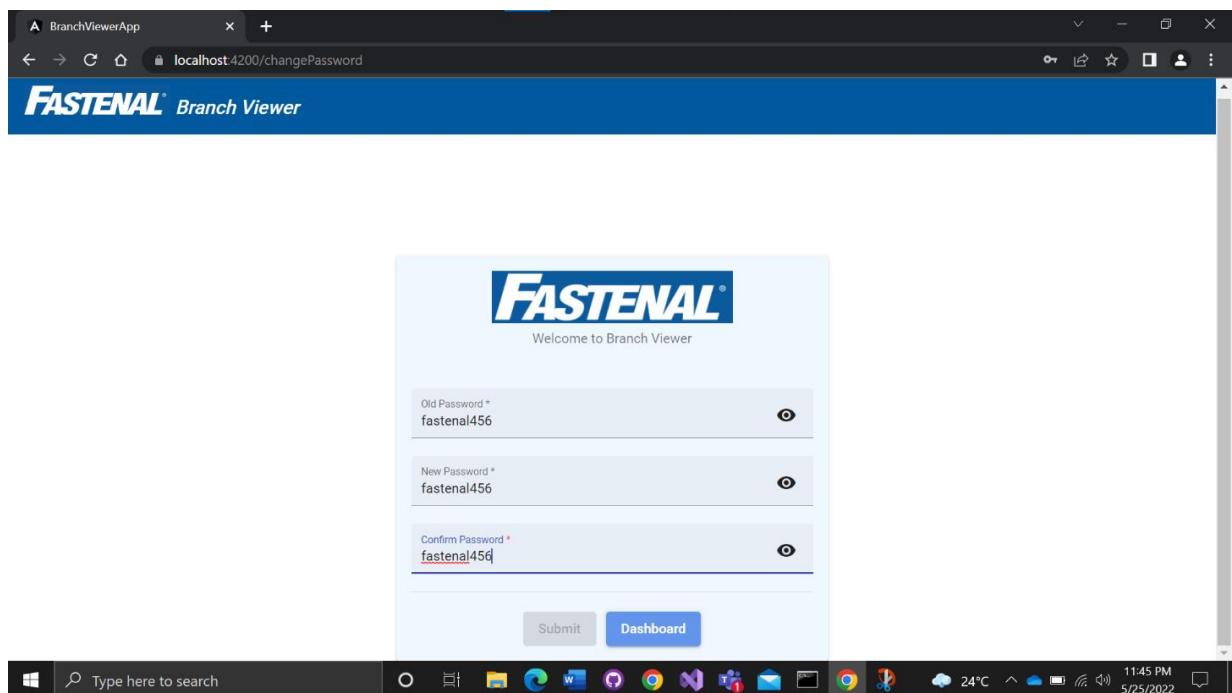


Figure 4.9 - Password Validation

Submit button will not be enabled if the values in the old password and the new password fields are the same.

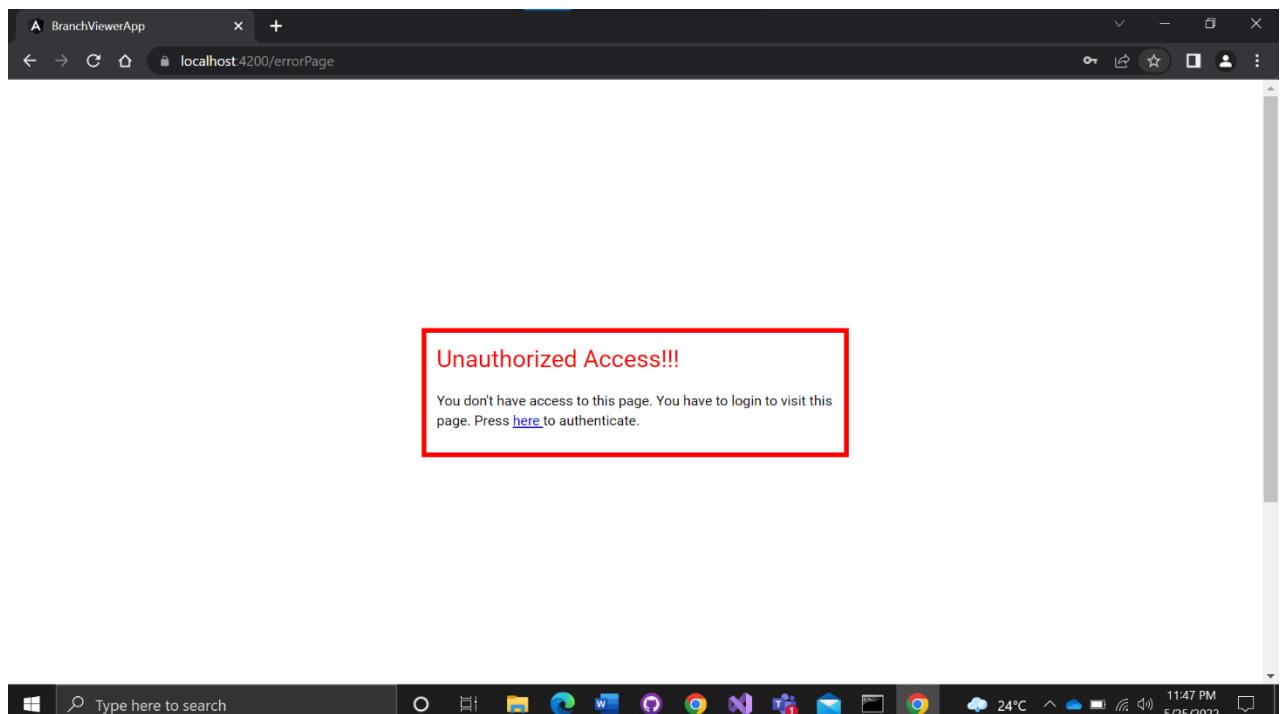


Figure 4.10 - Unauthorized Screen

This page will be loaded if the user tries to enter any page without logging in.

4.3 Significance of the results obtained

Every customer of Fastenal is associated with a branch where they visit often and is referred to home branch. This website will help the customers to list the branches (stores) of Fastenal, which is in their home state.

It also allows the users of this website to look the locations of the branches 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.

Route 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 MongoDB 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 branches.

- 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: Branch Location Manager 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.

5.3 Future Scope

5.3.1 Design 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, its address and other details which a user view upon clicking on any branch in the list view of the branches.

5.3.2 Server-Side Pagination

Another change that can be implemented is the server-side pagination with sorting and filtering features. Currently the type of pagination implemented in the application is the server-side pagination, but without filtering and sorting of the whole data. The server-side pagination with sorting and filtering functionalities can be implemented in the future version of the application as it is best for the larger data set. It is faster than the client-side pagination and more accessible as well.

5.3.3 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 branches. When a user clicks on any marker of the branch on the map, an option to provide navigation directions from the user's current location to that branch can be provided.

REFERENCES

Web

- [1] [Low Level Design Template](#)
- [2] [MongoDB Reference Best Practices](#)
- [3] [Angular Google Maps Components \(angular-maps.com\)](#)
- [4] [Mongo DB schema design](#)

PROJECT DETAILS

<i>Student Details</i>			
Student Name	Ghanashyam R K P		
Register Number	180905042	Section / Roll No	A / 08
Email Address	ghanashyamrkp@gmail.com	Phone No (M)	7538812077
<i>Project Details</i>			
Project Title	Fastenal: Branch Location Manager		
Project Duration	5 Months	Date of reporting	03/01/2022
<i>Organization Details</i>			
Organization Name	Fastenal India Sourcing, IT & Procurement Pvt Ltd		
Full postal address with pin code	3rd Floor, Tower B, Global Technology Park, Marathahalli Outer Ring Road, Devarabisanahalli, Bangalore - 560103		
Website address	fastenal.co.in.		
<i>External Guide Details</i>			
Name of the Guide	Vamsi Krishna Reddy Pallamala		
Designation	IT Associate Manager		
Full contact address with pin code	3rd Floor, Tower B, Global Technology Park, Marathahalli Outer Ring Road, Devarabisanahalli, Bangalore - 560103		
Email address	vpallama@fastenal.com	Phone No (M)	+91 80952 90909
<i>Internal Guide Details</i>			
Faculty Name	Dinesh Acharya		
Full contact address with pin code	Dept of Computer Science & Engg, Manipal Institute of Technology, Manipal.		
Email address	dinesh.acharya@manipal.edu		

PLAGIARISM REPORT