



## **Multi end customer management system ( CMS)**

**Project Guide : Mr. Sanket Jadhav**

### **Team Members:**

1. Tejas Thawari
2. Aniket Sawant
3. Nikhil Marathe
4. Manav Bhosale
5. Atul Kshirsagar

## Contents

Project Statement:	3
Acknowledgement	3
Introduction	4
Software Requirements Specifications	5
Required Technologies	6
1. React Js,	
2. Material UI	
3. Spring boot framework	
4. MongoDB	
5. Postman API platform	
Project Working	8
Implementation	10

**Project Statement :** Company needs to manage and keep the track of the business leads, data and information regarding it.

## Acknowledgement

We sincerely thank to all the persons whoever played a vital role in the successful completion of our project under titled Multi end customer management system ( CMS). We are grateful to Mr Sanket Jadhav under whose guidance we completed our project, who devoted their precious time for us. In spite of their busy schedules they always came forward to guide us in our work whenever needed. We are also thankful to Mr. Yash Salunkhe for his constant support.

In this hard time of Corona period, where the people are losing their jobs and business, the major losses are suffered by the young generation students, who are not unable to get internships and job placements opportunities. CodeKul Pvt Ltd came forward and provided so many free internships to students for their better future.

The Codekul organisation had arranged a free internship program of two months of computer courses. The courses offered in these internships are Web development, Python, Java, Artificial Intelligence, Machine Learning and many more. Firstly, they had taken interviews of the students who had enrolled in this internship program through Google meet. And then they had divided the selected students into various groups of 5 students each.

Our internship program begins from 12<sup>th</sup> June 2021. We, the group of 5 students, are developing our project in software development, Our topic name is Multi end customer management system ( CMS)

## Introduction

### What is a Customer Management System?

Customer Data is a precious asset to any business. A team that has clean, accurate data that is correctly formatted will be able to provide an appropriate level of service while saving time and money.

A customer management system is a cluster of all the systems, processes, and applications that are needed to manage customer relationships

We use React.js, Material UI for front end and Spring boot framework, mongoDB Atlas database, postman API platform for backend. To work on these Technologies we use different software applications. The CRM is a new technique in marketing and management where the owner tries to develop long term relationships with the clients and customers to develop them as lifetime customers. CRM aims to make the customer climb up the ladder of loyalty.

### Key Features of Our Project:

In our project, we had focused on some particular key points:

- Update information and data regarding all enquiries and leads
- Dynamic Admin Dashboard
- User Authentication
- Follow up dashboard

## Software Requirements Specifications

**Scope of work** – Multi end customer management system (CMS)

**Project Statement** - Company needs to manage and keep the track of the business leads, data and information regarding it.

### Users of System

1. Admin
2. Project Manager

### Scope

- Company will have a system that has information and data regarding all enquiries and leads.
- Admin will add the data of each and all enquiries and leads.
- Admin will have a dashboard to see all the stats and data of enquiries and leads till the current day.
- System will have the search option where the admin will be able to search the leads/enquiries data by Name, by date range too.
- There will be another follow up dash-board, to see the data of leads which are in pending & follow-ups are required.
- Each lead created will have a separate specific follow up system.
- For specific leads follow up will have all details of lead and Tasks and comments about follow up.
- All the leads created, tasks, as well as comments created, will be in descending order i.e the latest one should be visible on top.

## Required Technologies

6. React.Js,
7. Material UI
8. Spring boot framework
9. MongoDB
10. Postman API platform

### 1. ReactJS

- A open-source front-end JavaScript library for building user interfaces or UI components
- We use ReactJS to build a new single-page application. We made a login form, registration form, crud application and Admin Dashboard.
- Features
  1. JSX – JSX is a JavaScript syntax extension. It isn't necessary to use JSX in React development, but it is recommended.
  2. Components – React is all about components. You need to think of everything as a component. This will help you maintain the code when working on larger scale projects.
  3. Unidirectional data flow and Flux – React implements one-way data flow which makes it easy to reason about your app. Flux is a pattern that helps keep your data unidirectional.

### 2. Material UI

- a component library for React teeming with powerful components
- It is a most powerful and efficient tool to build an Application by adding Designs and Animations

### 3. Spring boot framework

- An application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE

- It provides flexible XML configurations, robust batch processing, database transactions, easy workflow, along with a wide variety of tools for development.

#### 4. MongoDB

- An object-oriented, simple, dynamic, and scalable NoSQL database. It is based on the NoSQL document store model. The data objects are stored as separate documents inside a collection — instead of storing the data into the columns and rows of a traditional relational database.
- These are dependencies of our project. We need to configure the following dependencies for MongoDB
  - The *spring-boot-starter-web* dependency provides the dependencies of a web application.
  - The *spring-data-mongodb* dependency provides integration with the MongoDB document database.

#### 5. Postman API platform

- A collaboration platform for API development. Postman's features simplify each step of building an API
- It sends the request to the server and receives the response back from the server
- We used a postman for Authorization. To Ensure that clients access data securely.

Requests. ... Organize your API requests into folders and groupings.

Examples. ...Run requests and collections against different data sets.

## Project Working

We made a website that has information and data regarding all enquiries and leads, we separated admin and users. There is a login form and registration panel, users need to add required information after successfully validating users can log in to the website. admin can add the data of each and all enquiries and leads. It has a dashboard to see all the flow and data of enquiries and leads till the current day. It has another follow up dash-board, to see the data of leads which are in pending and follow-ups are required. Each lead created will have a separate specific follow up system. All the leads created, tasks, as well as comments created, will be in descending order i.e the latest one should be visible on top.

It has two sides from end and backend, basically front-end refers to the user interface, while back-end means the server, application and database that work behind the scenes to deliver information to the user.

### Front end:-

Front end development which is created in Reactjs with help of many inbuilt libraries.

First of all we created a simple CRUD application then frontend of the admin dashboard. The user admin dashboard has many features like adding new users and showing previous entries for further follow up. We have used material-ui for the frontend part which makes react easy to use. We have used axios to communicate with the backend and it is a library which is used to make requests to an API, return data from the API, and then do things with that data in our React application that helps to integrate the front end with the backend with the help of localhost. React router is another package that is used for routing between the pages.

### Back end :-

A customer list – We will be creating a collection in our MongoDB database.

A list where customer query will be stored by admin/moderator A user can call the REST API to retrieve the customer query.

A user can call the REST API to retrieve all the query from the list POST – /v1/Employees– To add a customer query to list

GET – /v1/Employees – To retrieve all customer query from the list



GET – /v1/Employees/id – To retrieve a specific query

DELETE – /v1/Employees/id – To remove a query from the list

Users can sign up for a new account, or login with username & password.

By User's role (admin, moderator, user), we authorize the User to access resources

(role-based Authorization)

Methods	Urls	Actions
POST	/api/auth/signup	signup new account
POST	/api/auth/signin	login an account
GET	/api/test/all	retrieve public content
GET	/api/test/user	access User's content
GET	/api/test/mod	access Moderator's content
GET	/api/test/admin	access Admin's content

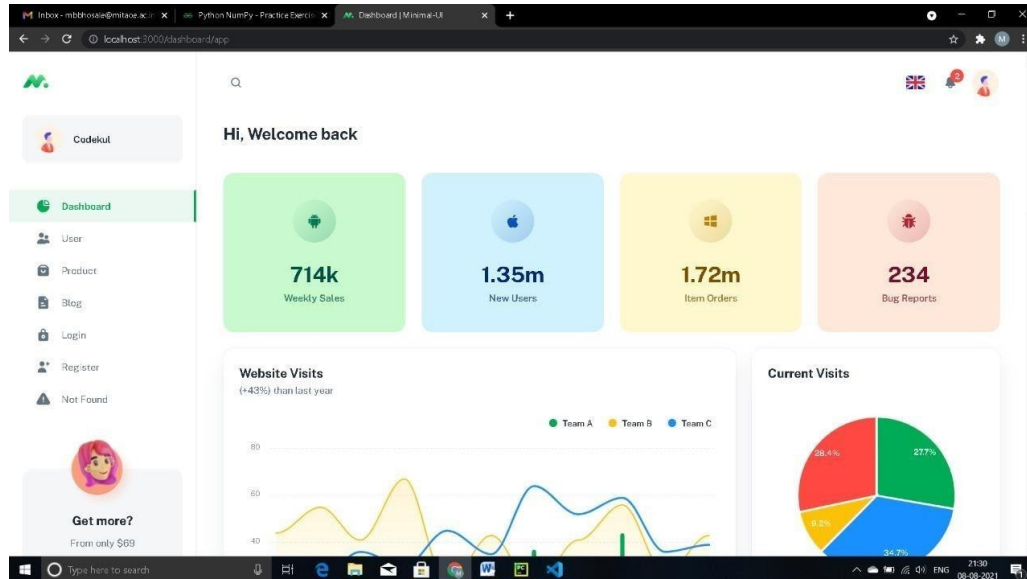
Spring Security will manage cors, csrf, session, rules for protected resources, authentication & authorization along with exception handler

The database we will use is MongoDB which can be accessed by the help of Spring Data

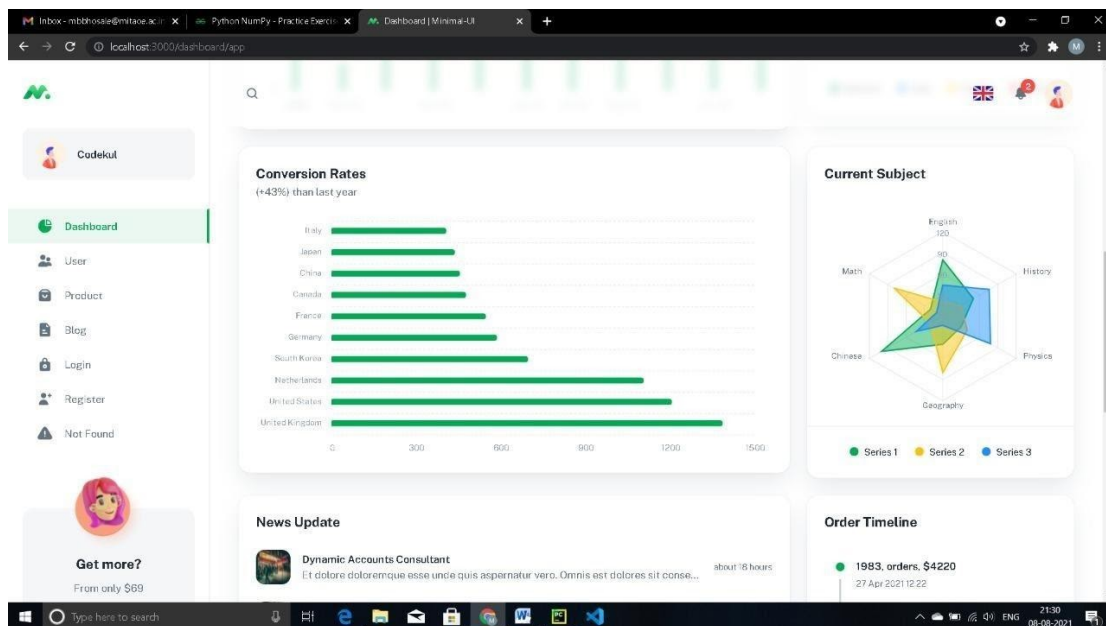
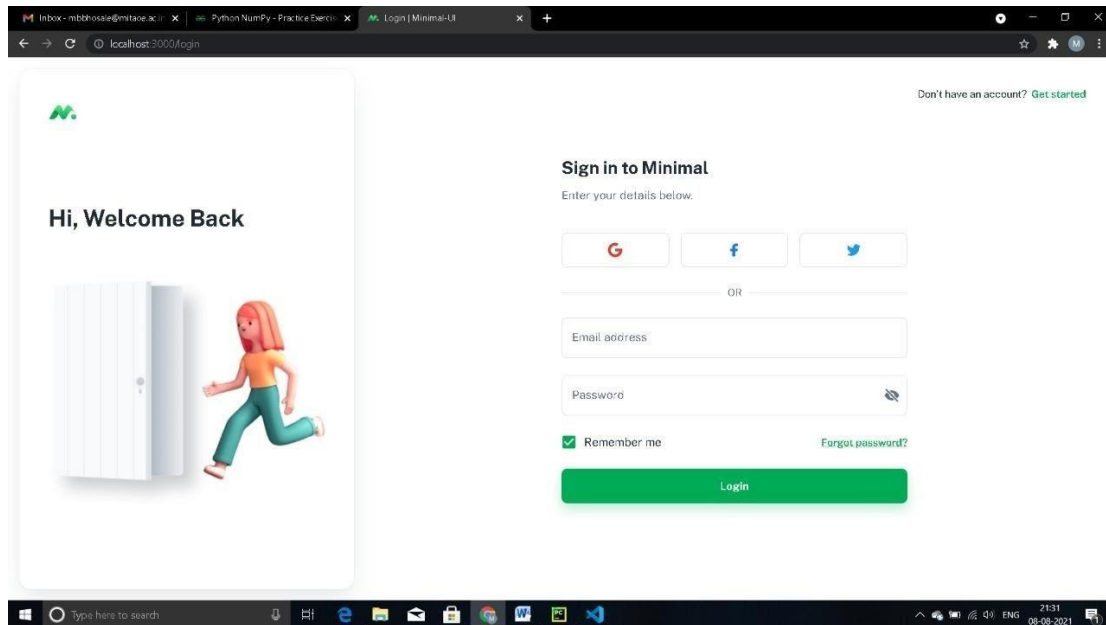
MongoDB.

# Implementation

## Front end Implementation



The screenshot shows a web registration form titled "Minimal" running on a browser at `localhost:3000/register`. The form includes a sidebar with a logo and the text "Manage the job more effectively with Minimal". The main content area features a registration form with the following fields: "First name", "Last name", "Email address", and "Password". There are also social media login buttons for Google, Facebook, and Twitter. A green "Register" button is at the bottom. The form also includes a link for "Already have an account? Login" and a note: "Get started absolutely free. Free forever. No credit card needed." Below the form, there is a link to the "Terms of Service and Privacy Policy".












## Back end Implementation

DOCUMENTS 3 AGGREGATIONS 0 SCHEMA 1 EXPLAIN PLAN 0 INDEXES 3 VALIDATION 0

FILTER { field: 'value' } OPTIONS FIND RESET ↺ ...

ADD DATA VIEW ( )

Displaying documents 1 - 3 of 3 C REFRESH

#	users	_id ObjectId	username String	email String	password String	roles Array	_class String
1		61952e5c87498a79165c4222	"Tejas"	"tejas@gmail.com"	"\$2a\$10\$ZHyd.419Ka86dRHNisYF6M"	[] 2 elements	"com.bezkoder.spring"   
2		6195392d902750171f9643d8	"Nikhil"	"Nikhil@gmail.com"	"\$2a\$10\$naC98kmlzrBdAJ08pMsoj.T"	[] 1 elements	"com.bezkoder.spring"   
3		610ba47ff59d1b6393910c69	"Aniket"	"Aniket@gmail.com"	"\$2a\$10\$07qlairprum8dxlnHpkL606"	[] 1 elements	"com.bezkoder.spring"   

### employeeDatabase.Employee










DOCUMENTS 3 TOTAL SIZE 912B AVG. SIZE 304B INDEXES 3 TOTAL SIZE 104.0KB AVG. SIZE 34.7KB



DOCUMENTS 3 AGGREGATIONS 0 SCHEMA 1 EXPLAIN PLAN 0 INDEXES 3 VALIDATION 0

FILTER { field: 'value' } OPTIONS FIND RESET ↺ ...

ADD DATA VIEW ( )

Displaying documents 1 - 3 of 3 C REFRESH

#	Employee	_id Int64	firstName String	lastName String	phoneNum String	location String	technology String
1		7	"Tejas"	"Thawari"	"21123"	"ghatanji"	"java"   
2		8	"nikhil"	"Marathe"	"211236"	"Yavatmal"	"java"   
3		11	"aniket"	"sawant"	"2112364"	"Yavatmal"	"java"   

http://localhost:8080/api/v1/employees Save   </>

POST http://localhost:8080/api/v1/employees Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1  {
2    ...
3    "firstName": "aniket",
4    "lastName": "sawant",
5    "emailId": "aniket@gmail.com",
6    "phoneNum": "2112364",
7    "location": "Yavatmal",
8    "technology": "java",
9    "status": "working",
10   "enquirySource": "codekul",
11   "enquiryDateCreated": "2012-10-24",
12   "nextFollowUpDate": "2012-01-02"
13 }
14

```

The screenshot displays a REST client interface with a request and response view. The request is a GET to `http://localhost:8080/api/v1/employees/7`. The response is a JSON object representing an employee.

**Request:**

- Method: GET
- URL: `http://localhost:8080/api/v1/employees/7`
- Body: This request does not have a body

**Response:**

- Status: 200 OK
- Time: 58 ms
- Size: 348 B

**Response Body (JSON):**

```
1 {
2   "id": 7,
3   "firstName": "Tejas",
4   "lastName": "Thawari",
5   "location": "ghatanji",
6   "technology": "java",
7   "enquirySource": "codekul",
8   "status": "working",
9   "enquiryDateCreated": null,
10  "nextFollowUpDate": null,
11  "emailId": "tsthawari@gmail.com"
12 }
```