Project Design Phase-2

CRM APPLICATION REQUIREMENTS

| Date | 02 Nov 2023 |
| --- | --- |
| Team ID | NM2023TMID02425 |
| Project Name | Application to make the Gas filling Station easy using CRM |

1.Open-Source Frameworks:

a) Web Framework:

You can use a web application framework to build the frontend and backend of your CRM. Some popular open-source option

i) Django (Python)

ii) Ruby on Rails (Ruby)

iii) Express.js (Node.js)

iv) Laravel (PHP)

b) Frontend Framework:

For the user interface, you can use open-source frontend frameworks like

i) React

ii) Vue.js

iii) Angular

iv) Bootstrap (for responsive design)

c) Database Management:

Utilize open-source database systems for data storage

i) PostgreSQL

ii) MySQL

iii) MongoDB (for NoSQL data)

2.Third-Party APIs:

a) Mapping and Location Services:

Integrate mapping and location services for helping users find gas stations. You can use APIs like Google Maps API or Map box.

b) Payment Gateway:

Incorporate a payment gateway for processing transactions. Common options include Stripe, PayPal, and Square.

c) SMS/Email Notifications:

Use third-party APIs for sending SMS and email notifications. Twilio and SendGrid are popular choices

d) Weather Services:

Integrate weather data to provide real-time weather updates to users. Services like Open Weather Map provide APIs for this purpose.

e) Queue Management:

If you want to include queue management functionality, consider using APIs like Qudini or Wait while.

f) Security and Authentication:

Implement secure user authentication using third-party authentication providers like Auth0 or Firebase Authentication.

3.Cloud Deployment:

a) Cloud Platform:

Choose a cloud service provider for hosting your application and data. Some popular options include:

1) Amazon Web Services (AWS)

2) Google Cloud Platform (GCP)

3) Microsoft Azure

4) Heroku (for simplified deployment)

b) Containerization:

Containerization with Docker and container orchestration with Kubernetes can help manage your application's scalability and deployment.

c) Serverless:

Consider using serverless computing for specific functions or microservices within your application. AWS Lambda and Google Cloud Functions are good options.

d) Database as a Service (DBaaS):

Leverage DBaaS solutions for database management and scaling. AWS RDS, Azure SQL Database, and Google Cloud SQL are examples.