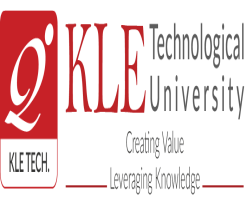
KLE Society's

KLE Technological University



**A Mini Project Report**

**On**

**CUSTOMER** **MANAGEMENT SYSTEM**

submitted in partial fulfillment of the requirement for the degree of

Master of Computer Applications

**Submitted By**

### Names of Students SRN

### 

Kavya Goudar 01FM19MCA025

Priyanka Rokhade 01FM19MCA014

Prarthana Charankar 01FM19MCA024

**Under the guidance of**

**Prakash Patil**

**Ashok Chikaraddi**

MASTER OF COMPUTER APPLICATIONS

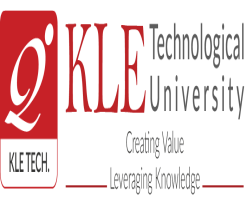
HUBLI – 580 031 (India).

Academic year 2020-21

KLE Society's

KLE Technological University

2020 - 2021



MASTER OF COMPUTER APPLICATIONS

**Certificate**

This is to certify that Minor Project entitled CUSTOMER **MANAGEMENT SYSTEM** is a bonafied work carried out by the student team **Ms.Kavya Goudar–01FM19MCA025**, **Ms. Priyanka Rokhade – 01FM19MCA014** , **Ms. Prarthana Charankar** **– 01FM19MCA024** in partial fulfillment of completion of Fourth semester Master of Computer Applications during the year 2019 – 2020. The project report has been approved as it satisfies the academic requirement with respect to the project work prescribed for the above said program.

**Prakash Patil**

**Ashok Chikaraddi Prakash Patil**

**(Guide Name) Head- MCA**

**External Viva:**

**Name of the Examiners Signature with date**

**1. Kavya Goudar**

**2 Priyanka Rokhade**

**3. Prarthana Charankar**

**CUSTOMER MANAGEMENT SYSTEM**

**INDEX**

Content Page No

1. Introduction……………………………………………………..5-6

a. Problem Statement

b. Objectives

c. Scope

1. Literature Survey……………………………………………….8-10
2. Existing System
3. Proposed System

1. Requirement Specification……………………………………..10-15
2. Functional Requirements
3. Non-Functional Requirements
4. Design…………………………………………………………..15-22
5. Architectural Design
6. Use Case Diagram
7. Data Flow Diagram
8. Class Diagram
9. ER Diagram
10. Screen Shots…………………………………………………….23-25
11. Code Snippets…………………………………………………...26-47
12. Schema Files
13. Model Files
14. Controller Files
15. Blade Files
16. Conclusion and Future Scope……………………………………47

1. References/bibliography…………………………………………48

**CUSTOMER MANAGEMENT SYSTEM**

**INTRODUCTION**

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. Storing customer data on emails, spreadsheets or paper is a task in itself. It is very tedious to retrieve data from such sources, and there are high chances of this information to be misplaced or deleted because it is unorganized. That’s when a Customer Management System comes into the picture. A customer management system is a cluster of all the systems, processes, and applications that are needed to manage customer relationships. It is also commonly known as Customer Management System**.**

**PROBLEM STATEMENT**

Most of the vendors/suppliers are managing their customers information using traditional methods such as register books. With such method the vendors/suppliers

Will not be able to communicate with their customers in a proper manner and in time. They are not able to trade with their customers with different localities properly as a result of lack of time there would be loss in the business and lossy trust of the customers.

So to overcome this problem we need to propose a proper software system.

This must includes design and develop application for managing customer information and their location for supply of goods/items and fetch feedback about services provided by the vendors to the customers.

**OBJECTIVES**

Customer Management System, the technology, along with human resources of the company, enables the company to analyze the behavior of customers and their value. The main areas of focus are as the name suggests: customer,relationship , and the management of relationshipand the main objectives to implement Customer Management System in the business strategy are:

* To simplify marketing and sales process
* To make call centers more efficient
* To provide better customer service
* To discover new customers and increase customer revenue
* To cross sell products more effectively
* The Customer Management System processes should fully support the basic steps of customer life cycle. The basic steps are:
* Attracting present and new customers
* Acquiring new customers
* Serving the customers
* Finally, retaining the customers

**SCOPE**

Traditionally and essentially CRM is management software for sales, marketing and customer service teams as they are the major touch points for any customer contact strategy. Customer Management System for sales management

A good mobile enabled Customer Management System will allow sales reps to manage their tasks, activities and meetings from wherever they are, reducing unwanted administration time and building in best practice into prospect management. Sales managers who need real time transparency in reviewing their team’s sales pipeline and KPIs can do so with the use of configurable Customer Management System dashboards. When reviewing Customer Management System choices the sales functionality should cover the basics of:

* Sales force and pipeline management
* Lead, contact and prospect management
* Customer Management System for marketing

Customer Management System can provide both marketing and sales functionality within their own operational requirements; but should also enable better collaboration and transparency between teams; putting the lead, prospect and customer at the heart of the Customer Management System strategy.

Marketing teams can be supported in lead generation efforts, planning and executing multi-channel marketing campaigns, segment audiences, deliver targeted messaging at the right time, analyze marketing spend ROI and carry out best practice testing efficiently. Marketing functionality within a Customer Management System review should include:

* Multi-channel marketing campaign management
* Database management
* Social media engagement
* Customer Management System for customer service

Customer service teams need to be able to deliver a positive customer experience and to do this they need to be supported by a Customer Management System ,system that gives them a good record of past customer contacts, support enquiries, technical incidents and product history.

Customer service operators need to handle calls, emails and social/web enquiries quickly and to the satisfaction of the customer; but also keeping servicing KPIs such as response times, resolution percentages and SLA adherence.

A Customer Management System should provide a 360 degree view of the customer and support operatives with information they need at their fingertips so they can resolve issues and handle queries efficiently.

When considering Customer Management System functionality for customer service, organizations should review:

* Customer service and contact management
* Knowledge sharing and document management
* Computer Telephony Integration (CTI)
* SLA and contract management
* Social listening integration

Understanding the Customer Management System scope you hope to achieve is crucial to the success of your Customer Management System implementation. “Scope” refers to how far-reaching the Customer Management System will be within your organization. For example, if only your customer service reps will be using it, the scope of your Customer Management System would be considered narrow. If your marketing teams, sales reps, customer care team, data analysts and project managers will all be utilizing the system, that is a broad scope of Customer Management System.

To begin the scoping process, reflect on what you’re hoping to improve by utilizing a Customer Management System, system — this should help narrow your focus and help you understand what kind of scope you need. To keep expectations realistic, make a list of questions for yourself and for the vendors of your shortlist platforms. Consider things like budget, time-frame, how much training you think you’ll want (or, more likely, how much you’re willing to pay for). By asking these targeted questions to the project managers and stakeholders, you can move forward confidently with parameters that have been discussed with all the necessary people on board.

**LITERATURE SURVEY**

Review of Literature on CMS

Customer Relationship Management (CRM) has become one of the most dynamic technology topics of the millennium. According to Chen and Popovich (2003), CRM is not a concept that is really new but rather due to current development and advances in information and enterprise software technology, it has assumed practical importance. The root of CRM is relationship marketing, which has the objective of improving the long-term profitability of customers by moving away from product-centric marketing. Bose (2002) noted that CRM was invented because the customers differ in their preferences and purchasing habits . If all customers were alike, there will be little need for CRM. As a result, understanding customer drivers and customer profitability, firms can better tailor their offerings to maximize the overall value of their customer portfolio (Chen and Popovich) . The attention CRM is currently receiving across businesses is due to the fact that the marketing environment of today is highly saturated and more competitive (Chou et al, 2002) . According to Greenberg (2004), CRM generally is an enterprise-focused endeavor encompassing all departments in a business . He further explains that, in addition to customer service, CRM would also include, manufacturing, product testing, assembling as well as purchasing, and billing, and human resource, marketing, sales and engineering. Chen and Popovich (2003) argued that CRM is a complicated application which mines customer data, which has been retrieved from all the touch points of the customer, which then creates and enable the organization to have complete view of the customers. The result is that firms are able to uncover and determine the right type of customers and predicting trend of their future purchases. CRM is also defined as an all embracing approach that seamlessly integrates sales, customer service, marketing, field support and other functions that touch customers (Chou et al, 2002) . They further stated that CRM is a notion regarding how an organization can keep their most profitable customers and at the same time reduce cost, increase in values of interaction which then leads to high profits.

Benefits of CMS

According to Chen and Popovich (2003), CRM applications have the ability to deliver repositories of customer data at a much smaller cost than old network technologies. Throughout an organization, CRM systems can accumulate, store, maintain, and distribute customer knowledge. Peppard (2000) noted that effective management of information has a very important role to play in CRM because it can be used to for product tailoring, service innovation; consolidate views of customers, and for calculating customer lifetime value. CRM systems assists companies evaluate customer loyalty and profitability based on repeat purchases, the amount spent, and longevity. Bull (2003) added CRM makes it practicable for companies to find unprofitable customers that other companies have abandoned. This position is supported by Galbreth and Rogers (1999) that CRM helps a business organization to fully understand which customers are worthwhile to acquire , which to keep, which have untapped potential, which are strategic, which are important , profitable and which should be abandoned.

**EXISTING SYSTEM**

There are not any existing systems for client side. So all the work are handle manually and have to be noted down in some register and also taking care of that documentation. They are arranged meeting by call and if any update occurred then again call the client and update meeting schedule, its wasting time and as well as money also and also the disturb the valuable clients.

WHY THIS SYSTEM?

To overcome over problems they are thinking to create such a system for the client as well as Company use. In which we can silently telling the customer about their status and meeting schedule by mailing and sms. And they can also trace their payment and project detail. By this system company having the lots of advantages by having a bulk year records and all the data together.

**PROPOSED SYSTEM**

The Customer Management System is a new technique in marketing and management where the owner tries to develop long term relationship with the clients and customers to develop them as life time customers. CMS aims to make the customer climb up the ladder of loyalty. The company first tries to determine who are likely prospects i.e. the people who have a strong potential interest in the product and ability to pay for it. The company hopes to convert many of its qualified prospects into first time customers into lifetime clients. They are those people who buy only from the company in the relevant product categories. The next challenge for the company is to convert this client into advocates. Advocates are those clients who praise the company and encourage others to buy from it. The ultimate challenge is to convert these advocates into partners where the customers and the clients work actively together to discover ways of getting mutual benefit. Our purpose is to fulfill the following activities: 1. Client Solutions 2. Project Management 3. Quotations 4. Suggestion 5. Lead time management 6. on line tracking for particular project 7. On line payment detail 8. Reports by daily Emails / sms with status 9. Query Solve 10. Interact with the company's developers or marketing manager

**REQUIREMENT SPECIFICATION**

**FUNCTIONAL REQUIREMENTS**

The application contains the following core features for functionality:

* Register/Sign-up
* Sign-in
* Customer details addition
* Feedback submission
* Add location details

**Register/Sign-up and Sign-in**

This feature requires the user to install the application and register himself into the system. The first page of the application will lead the user to the registering form.

**Sign-in**

If the customer has already registered then he/she can sign-in directly. If not , he/she must register and then sign to access the “Customer Management System”.

**Customer details addition**

The customer once signs-in to the website he/she can add their details in the website.

**Feedback submission**

The customers are allowed to give feedback here in the website. He/she can submit feedback accordingly.

Here we are given with location addition option also. The customer can add their locations respectively.

**NON-FUNCTIONAL REQUIREMENTS**

**Performance Requirements**

To run the application, users will require an android oriented phone with minimum configuration of

* Laravel 4.5 or above.
* Strong Internet connection.
* 1Gb Storage Space
* Accessibility permissions for storage, Internet.
* XAMP OR WAMP for back-end server.
* A good text editor ex, atom or visual studio.

**Safety Requirements**

“Customer Management System” will not affect data stored outside of its servers nor will it affect any other applications of the user’s. It cannot cause any damage to user system or its internal components.

**Security Requirements**

This website assumes that only the user will have access to his/her account. With this being said, only a Google email address with password is required to verify the identity of the user upon opening the website.

**Software Quality Attributes**

The graphical user interface of “Customer Management System” website keeps usability as the first priority while designing. The website presented and organized in such a manner that is both visually appealing and easy for the user to navigate through the menu. There will be feedbacks and visual cues such as notifications to inform users of updates.

**Credentials**

Email id and password.

**Why You Need WAMP, MySQL, and PHP?**

PHP is a powerful scripting language that can be run by itself in the command line of any computer with PHP installed. However, PHP alone isn't enough in order to build dynamic web sites. To use PHP on a web site, you need a server that can process PHP scripts. WAMP server allows developers to test PHP scripts locally; this makes it an invaluable piece of your local development environment.

Additionally, dynamic websites are dependent on stored information that can be and easily; this is the main difference between a dynamic site and a static HTML site. However, PHP doesn’t provide a simple, efficient way to store data.

**PHP:**

PHP originally stood for “Personal Home Page” and was released as a free, open source project. Over time, the language was reworked to meet the needs of its users. It was originally created by Rasmus Lerdorf in 1994.In 1997, PHP was renamed to the current “PHP: Hypertext Preprocessor”. Hypertext Preprocessor (or simply PHP) is a general-purpose programming language originally designed for web development. PHP is generally used as a server-side scripting language; it is especially well-suited for creating dynamic web pages and client-side GUI applications. PHP generally runs on a web server, taking PHP code as its input and creating web pages as output The scripting language features integrated support for interfacing with databases such as MySQL, which makes it a prime candidate for building all manner of web applications, from simple personal web sites to complex enterprise-level applications.PHP is a scripted language, which is another great advantage for PHP programmers. It is available free of charge, and the PHP Group provides the complete source code for users to build, customize and extend for their own use.

**Usage:**

PHP is a general-purpose scripting language that is especially suited for web development. PHP generally runs on a web server, taking PHP code as its input and creating web pages as output. It can also be used for command-line scripting and client-side GUI applications. It is available free of charge, and the PHP Group provides the complete source code for users to build, customize and extend for their own use.

PHP primarily acts as a filter, taking input from a file or stream containing text and/or PHP instructions and outputs another stream of data; most commonly the output will be HTML. It can automatically detect the language of the user. Originally designed to create dynamic web pages, PHP’s principal focus is server side scripting, and it is similar to other server-side scripting languages that provide dynamic content from a web server to a client, such as Microsoft’s Active Server Pages, Sun Microsystems’ Java Server Pages, and mod\_perl. PHP has also attracted the development of many frameworks that provide building blocks and a design structure to promote rapid application development (RAD).

**HTML:**

Hyper Text Markup Language (HTML) is a markup language for creating a webpage. Web pages are usually viewed in a web browser. They can include writing, links, pictures, and even sound and video. HTML is used to mark and describe each of these kinds of content so the web browser can display them correctly. HTML can also be used to add meta information to a webpage. Meta information is usually not shown by web browsers and is data about the web page, e.g., the name of the person who created the page. Cascading Style Sheets (CSS) is used to style HTML elements while JavaScript is used to manipulate HTML elements and CSS styles.

The global publishing format of the Internet is HTML. It allows authors to use not only text but also format that text with headings, list and tables, and also includes still images videos, and sound within text. HTML pages can also be used for entering a data and as a front end for commercial transaction.

**MY SQL:**

SQL Server is a Relational Database Management System (RDBMS) that runs exclusively under the Windows operating system. One benefit of using Windows exclusively is that you can send and receive E-mail messages based on SQL Server "events" and you can also let the operating system handle login security. The data base is an organized collection of data. A database management system (DBMS) such as Access, FileMaker Pro, Oracle or SQL Server provides you with the software tools you need to organize that data in a flexible manner. It includes facilities to add, modify or delete data from the database, ask questions (or queries) about the data stored in the database and produce reports summarizing selected contents.MySQL is a multithreaded, multi-user SQL database management system (DBMS). The basic program runs as a server providing multi-user access to a number of databases.

**Security:**

View are basically used as a part of security, means in many organizations ,the end user will never be given original tables & all data entry will be done with the help of views only. But the data base administrator will be able to see everything because all the operations done by the different users will come to the same table.

**Laravel**

Laravel is a web application framework with expressive, elegant syntax. A web framework provides a structure and starting point for creating your application, allowing you to focus on creating something amazing while we sweat the details.

Laravel strives to provide an amazing developer experience, while providing powerful features such as thorough dependency injection, an expressive database abstraction layer, queues and scheduled jobs, unit and integration testing, and more.

Whether you are new to PHP or web frameworks or have years of experience, Laravel is a framework that can grow with you. We'll help you take your first steps as a web developer or give you a boost as you take your expertise to the next level. We can't wait to see what you build.

[**Why Laravel?**](https://laravel.com/docs/8.x#why-laravel)

There are a variety of tools and frameworks available to you when building a web application. However, we believe Laravel is the best choice for building modern, full-stack web applications.

**A Progressive Framework**

We like to call Laravel a "progressive" framework. By that, we mean that Laravel grows with you. If you're just taking your first steps into web development, Laravel's vast library of documentation, guides, and [video tutorials](https://laracasts.com/) will help you learn the ropes without becoming overwhelmed.

If you're a senior developer, Laravel gives you robust tools for [dependency injection](https://laravel.com/docs/8.x/container), [unit testing](https://laravel.com/docs/8.x/testing), [queues](https://laravel.com/docs/8.x/queues), [real-time events](https://laravel.com/docs/8.x/broadcasting), and more. Laravel is fine-tuned for building professional web applications and ready to handle enterprise work loads.

**A Scalable Framework**

Laravel is incredibly scalable. Thanks to the scaling-friendly nature of PHP and Laravel's built-in support for fast, distributed cache systems like Redis, horizontal scaling with Laravel is a breeze. In fact, Laravel applications have been easily scaled to handle hundreds of millions of requests per month.

Need extreme scaling? Platforms like [Laravel Vapor](https://vapor.laravel.com/) allow you to run your Laravel application at nearly limitless scale on AWS's latest serverless technology.

**A Community Framework**

Laravel combines the best packages in the PHP ecosystem to offer the most robust and developer friendly framework available. In addition, thousands of talented developers from around the world have [contributed to the framework](https://github.com/laravel/framework). Who knows, maybe you'll even become a Laravel contributor.

**4. DESIGN**

**ARCHITECTURAL DESIGN**

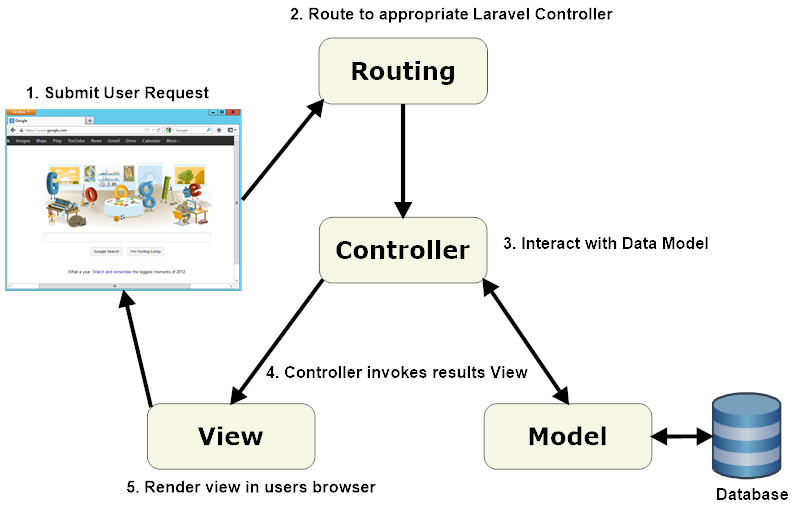
An early stage of the system design process.

• Represents the link between specification and design processes.

• Often carried out in parallel with some specification activities.

• It involves identifying major system components and their communications.

Simple, informal block diagrams showing entities and relationships are the most frequently used method for documenting software architectures.



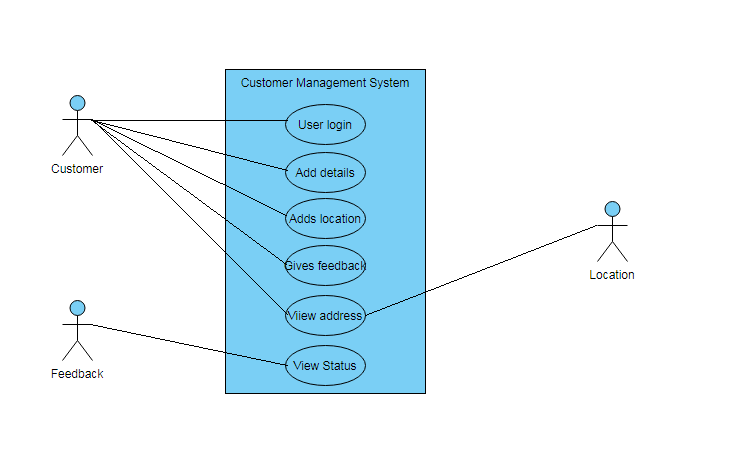
**USE CASE DIAGRAM**

Use case diagram is a graphic depiction of the interactions among the elements of customer management system.

It represents the methodology used in system analysis to identify, clarify and organize system requirements.

The main actors of CMS in this use case diagram are:

* Customer Entity: Use case of customer entity manages location, feedback and crud operations.
* Location Entity: Use case of location entity manages location details.
* Feedback Entity: Use case of feedback entity submits feedback and views status.



**DATA-FLOW DIAGRAM**

The data flow diagram (DFD) is one of the important modeling tools. It shows the user of the data pictorially. DFD represents the flow of the data between different transformations and processes in the system. The data flow diagram shows logical flow of the data. It represents the functional dependencies within a system. It shows output values in a computation are derived from input values. It is a simple pictorial representation or model for system behavior. It specifies, “What is to be done but not how is to be done”. It describes the logical structure of the system. It relates data information to various processes of the system. It follows top-down approach.

**Data Flow Diagram Notations:**

* **Data Flow:**

It may be from file-to-file or file-to-process or process- process. It is generally in terms of attributes. There may be either an input data flow or output data flow.

* **Functional Processing:**

The process is nothing but the transformation of data it starts with the subject and it has the verb followed by the subject.

* **Data store:**

It includes file, data base and repository. To parallel lines represent it or a one end closed rectangle

Or

* **Actor/source/sink:**

The files which are outside the system and used by the process or the processes of the system. Generally

Source/Sink in the actor.

**Objectives**

• To graphically document boundaries of a system.

• To provide hierarchy breakdown of the system.

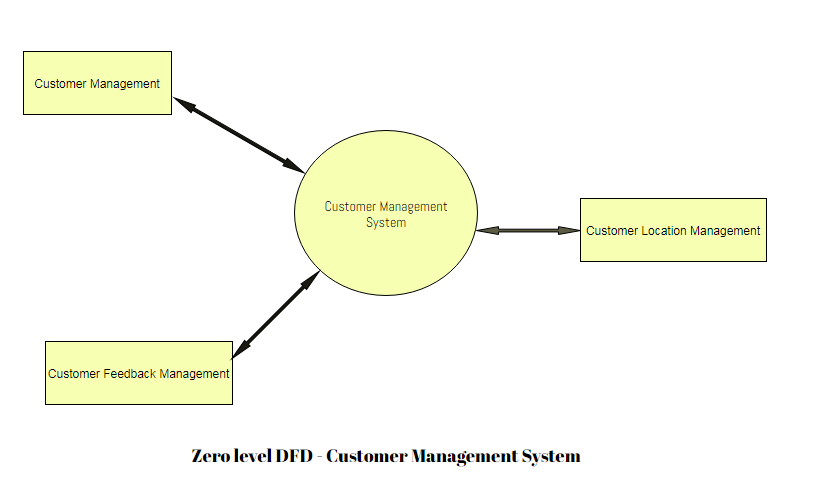
• To show movement of information between a system and its environment.

• To document information flows within the system.

• To aid communication between users and developers.

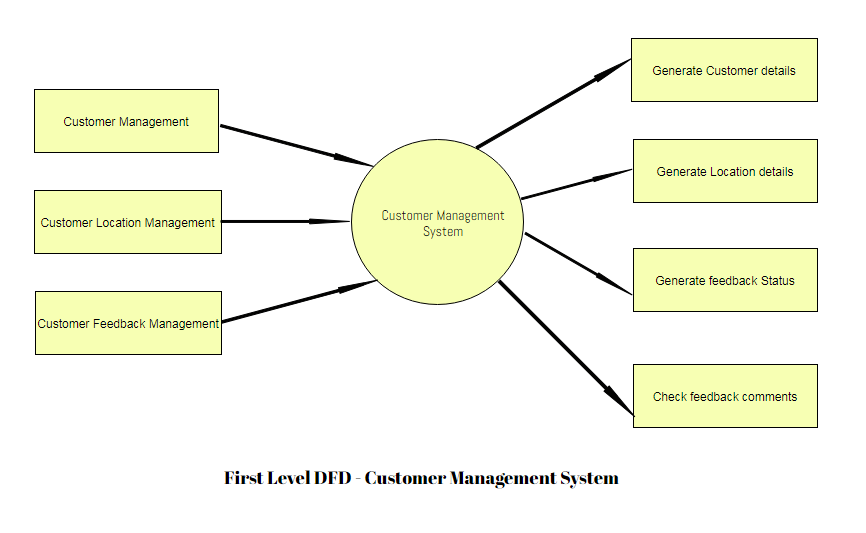
**ZERO – LEVEL DFD**

This is zero level DFD of customer management system where we elaborate high level process of a customer.It is designed for the view of feedback , location and customer details.



**LEVEL - ONE DFD**

Level one DFD of CMS shows how the system is divided into sub-systems (processes), each of which deals with one or more flows to our form an external agents, and which together provide all functionality of customer management system.



**CLASS DIAGRAM**

Customer Management System describes the structure of classes, their attributes, operations (or methods), and the relationships of objects.

The main classes of CMS are customer, location, feedback.

* Customer class: Manages all the operations of the customers.
* Location class: Manages all the operations of the locations.
* Feedback class: Manages all the operations of the feedbacks.
* Customer Attributes :

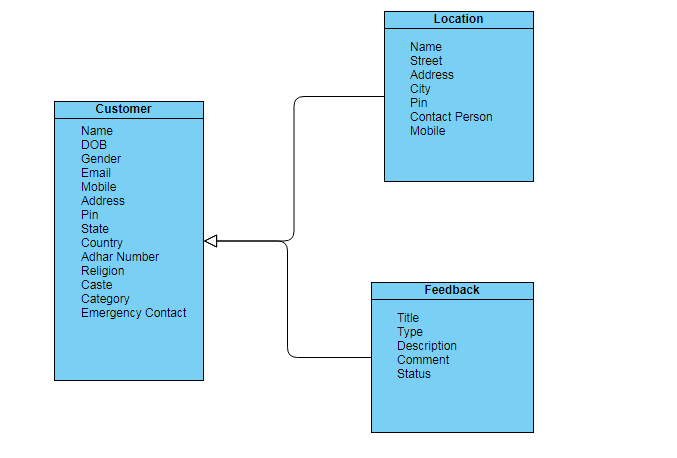
Name, Dob, Gender, Email, Mobile\_no, Address, City, Pin, State, Country, Aadhar\_no, Religion, Caste , Category, Emergency\_contacts.

* Location Attributes :

Name, Street, Address, City, Pin, Contact\_Person, Mobile.

* Feedback Attributes :

Title, Type, Description, Comments, Status.



**ER DIAGRAM**

In software engineering, an Entity-Relationship Model (ER model for short) is an abstract and conceptual representation of data. Entity-Relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion. Diagrams created by this process are called Entity-Relationship Diagrams or ER Diagrams.

An entity may be defined as a thing which is recognized as being capable of an independent existence and which can be uniquely identified. An entity may be a physical object such as a house or car, an event such as a house sale or a car service, or a concept such as customer transaction or order. A entity- type is a category. An entity, strictly speaking, is an instance of a given entity type. There are usually many instances of an entity –type.

The ER diagram represents the model of customer management system entity.

The Entity relationship diagram of CMS shows all the visual instrument of data base tables and the relations between customer, feedback and location.

It uses structure data and to define the relationships between structure data groups of CMS functionalities.

The main entities and their attributes are:

* Customer Attributes :

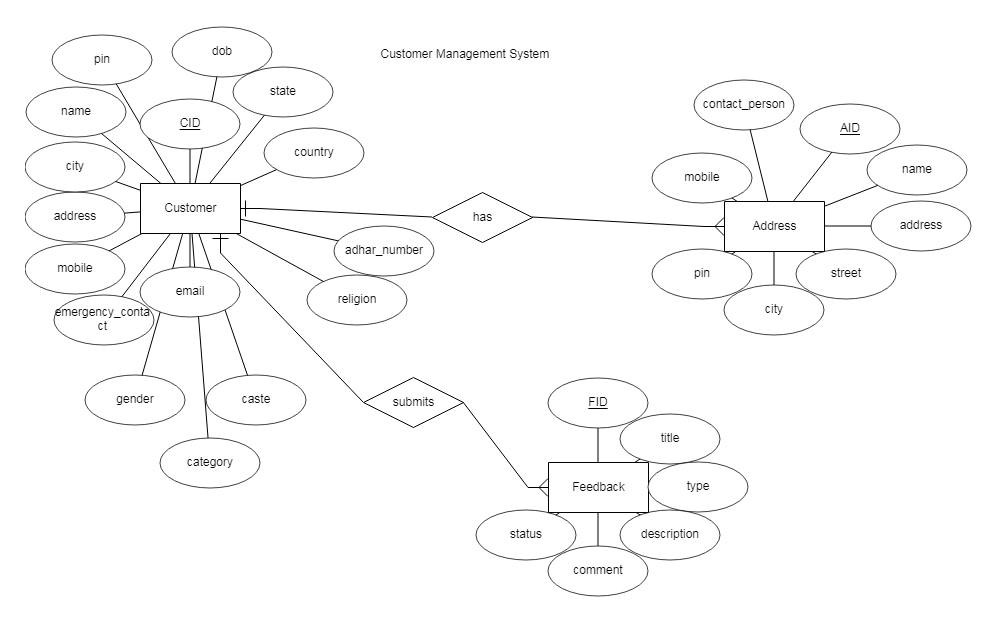
Name, Dob, Gender, Email, Mobile\_no, Address, City, Pin, State, Country, Aadhar\_no, Religion, Caste , Category, Emergency\_contacts.

* Location Attributes :

Name, Street, Address, City, Pin, Contact\_Person, Mobile.

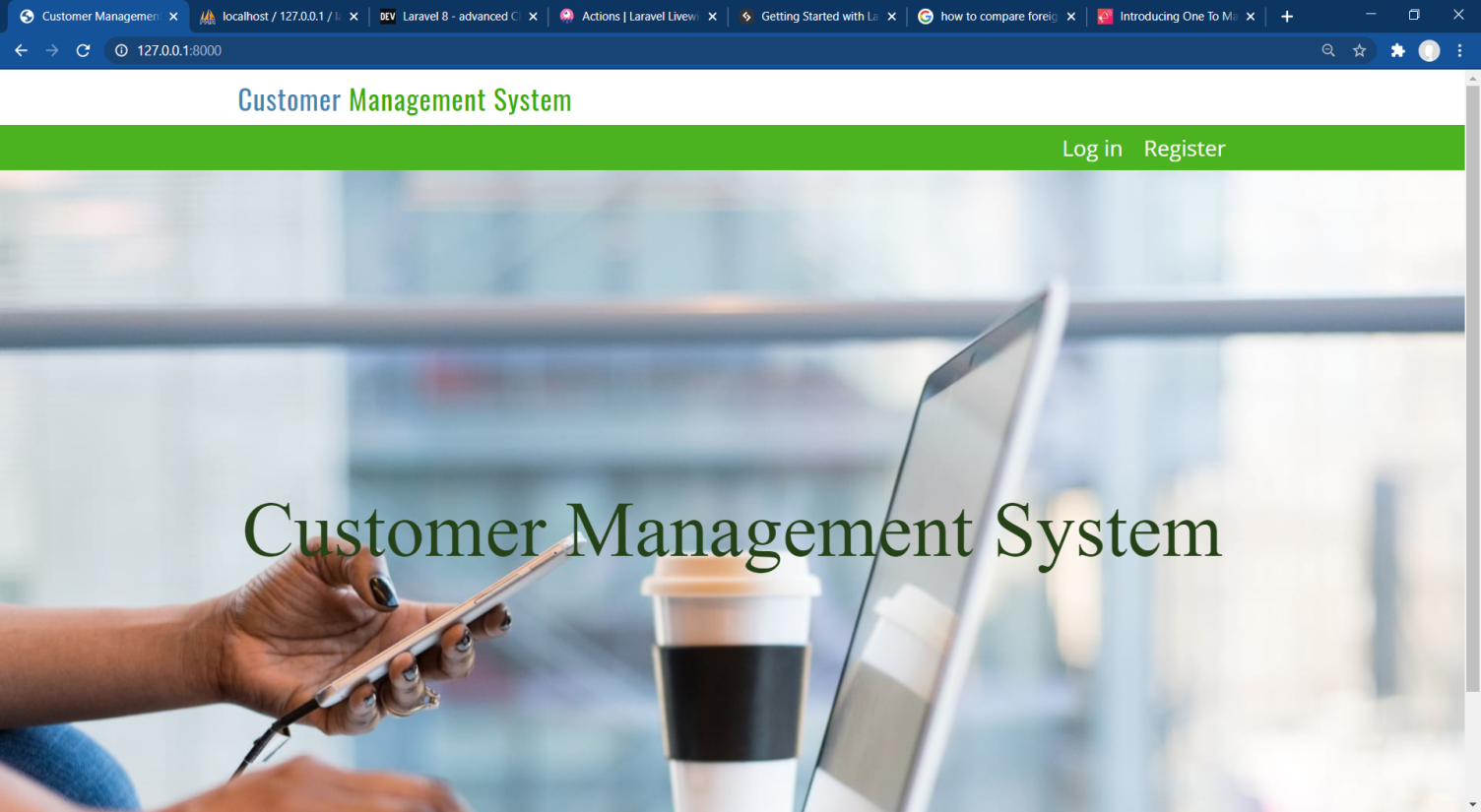
* Feedback Attributes :

Title, Type, Description, Comments, Status.

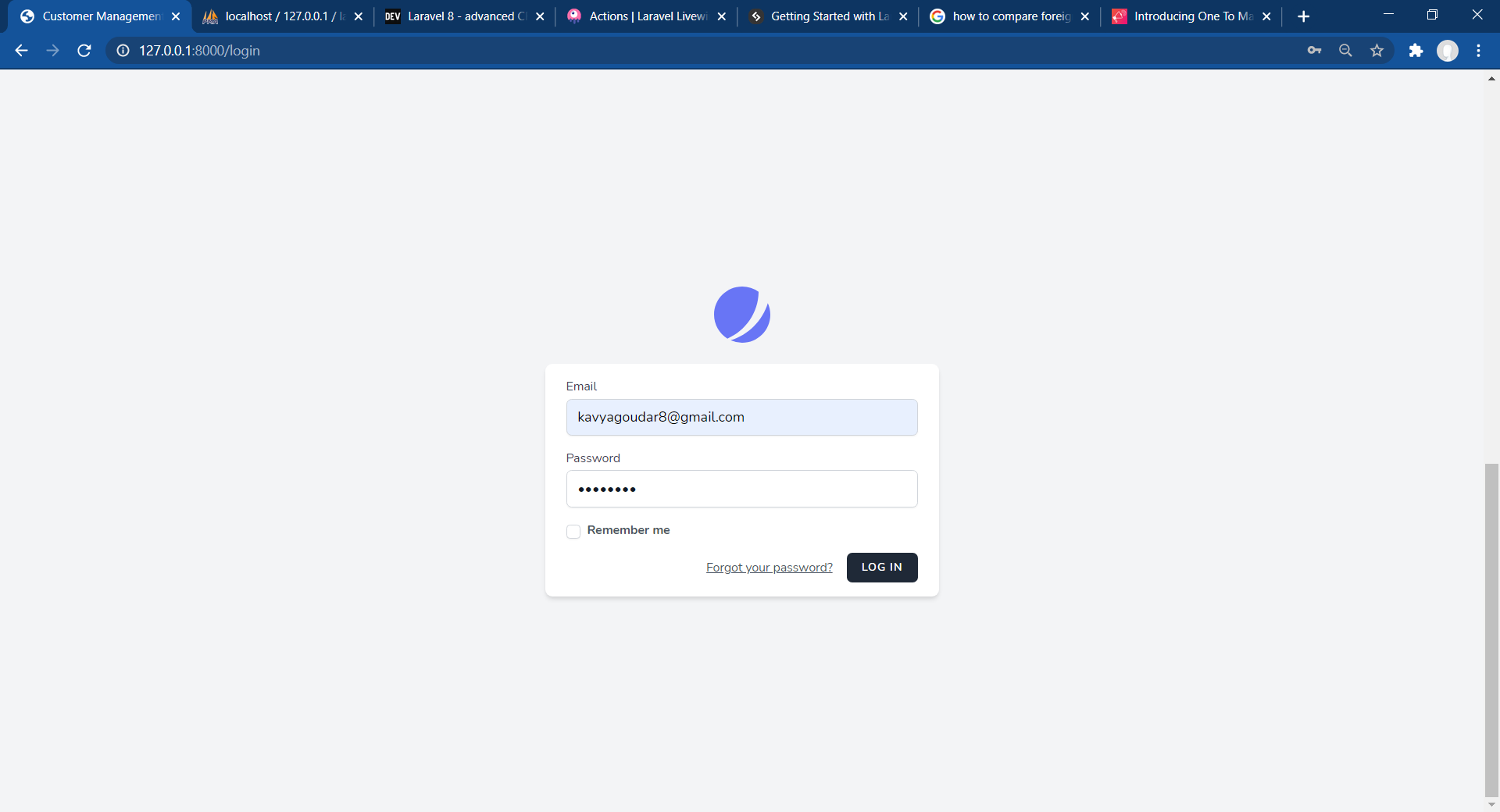


**Screen Shots of Customer Management System**

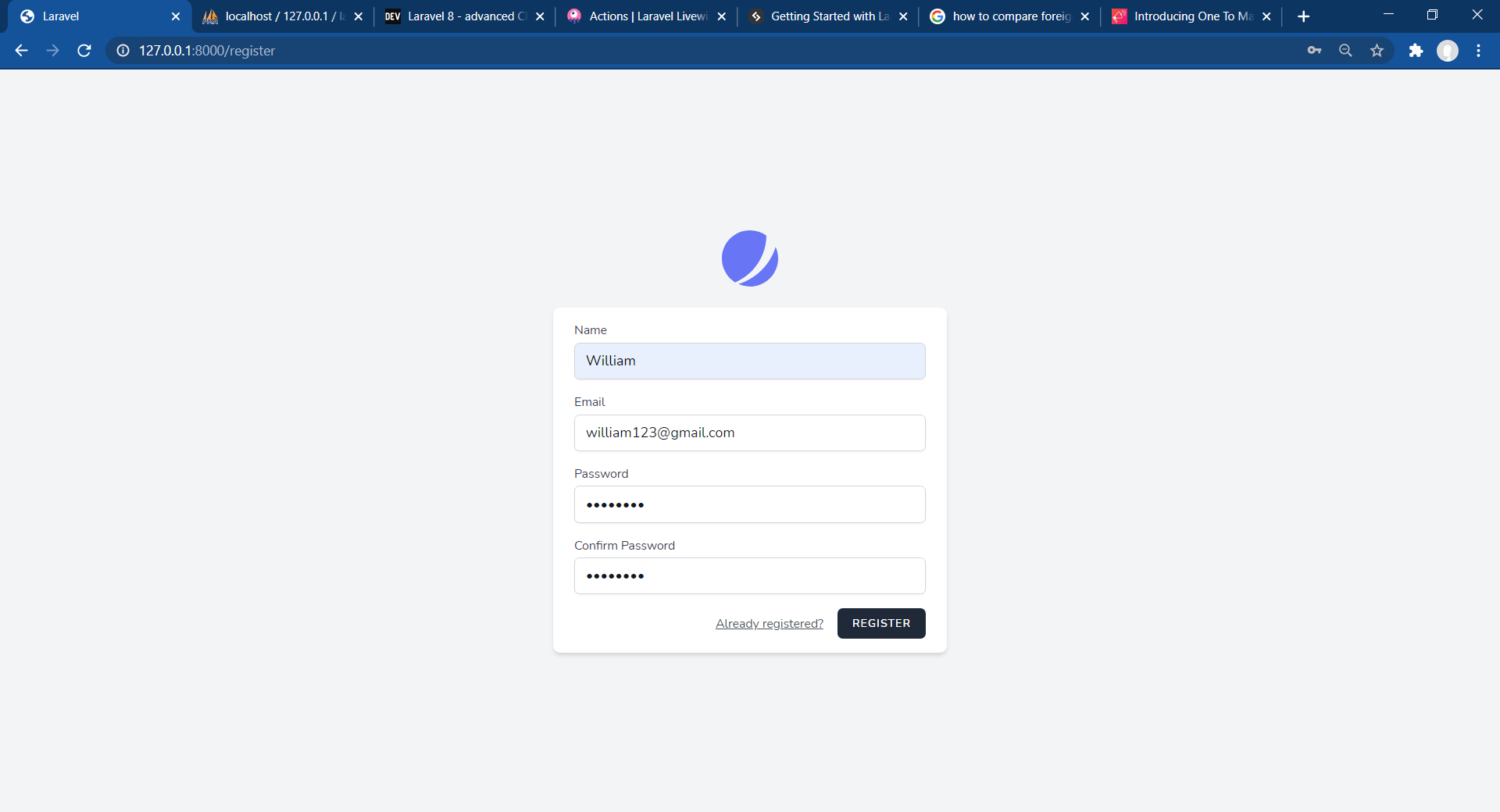
**Home Page**



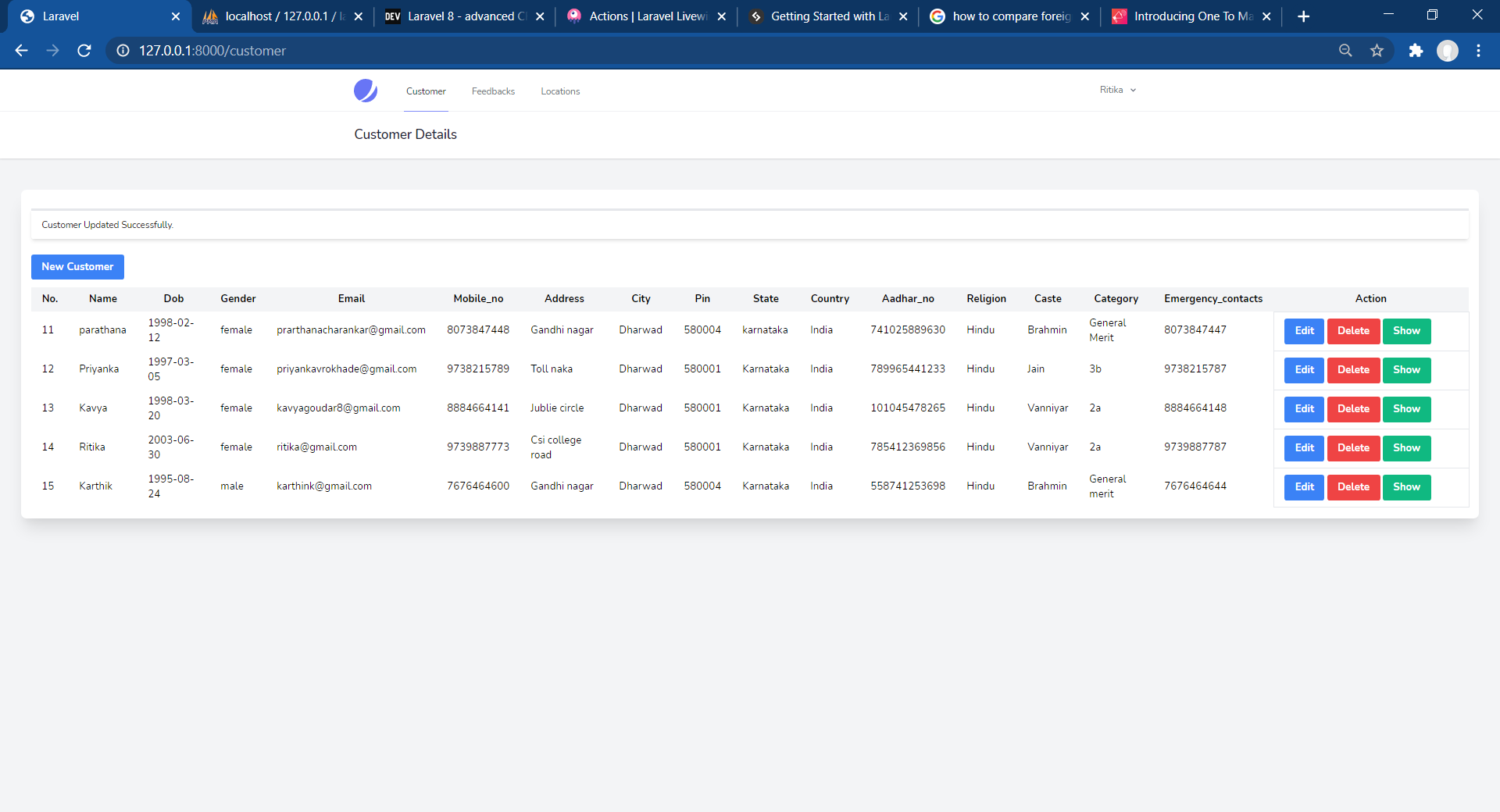
**Log-in Page**

******

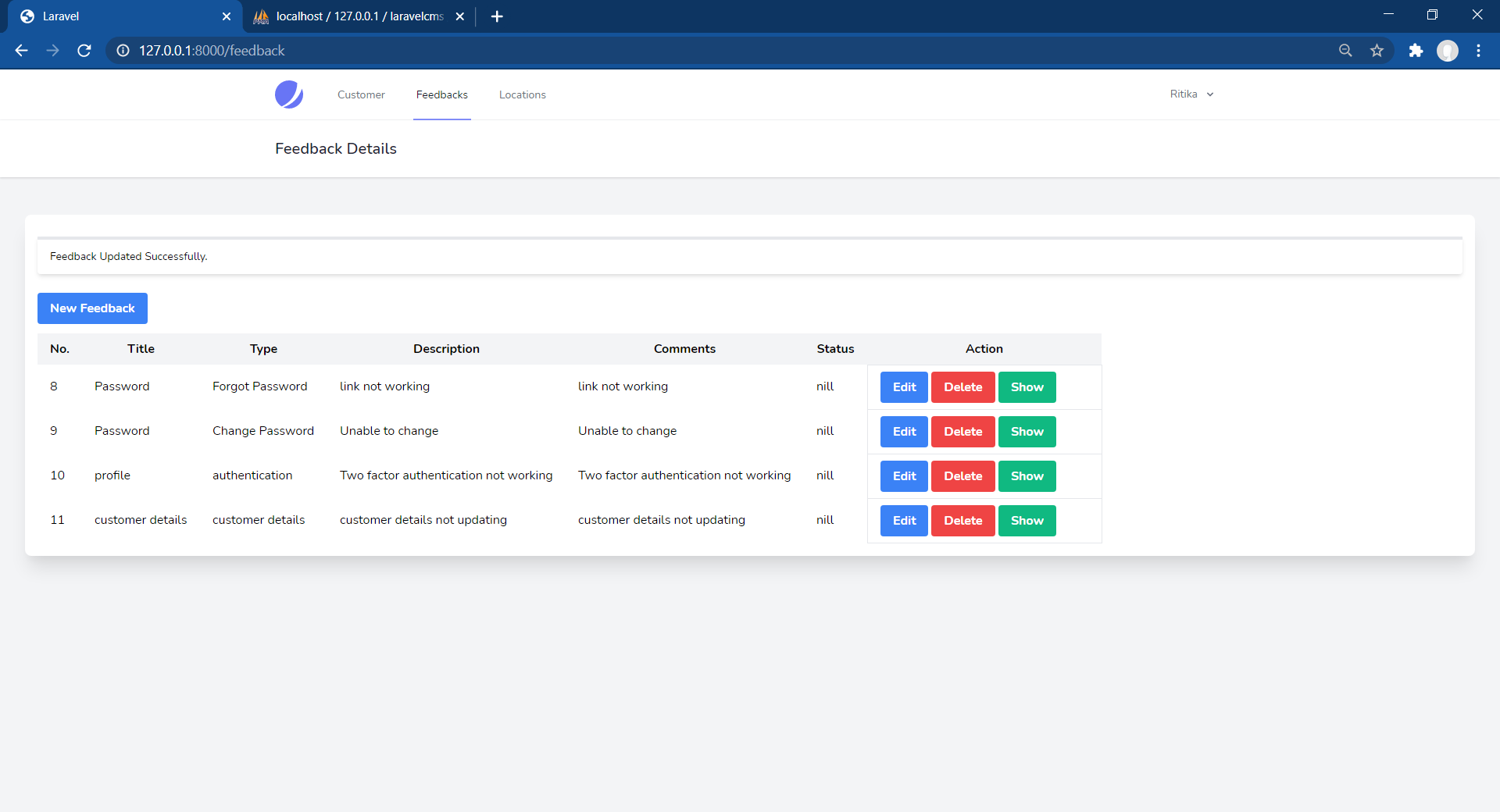
**Register - Page**

******

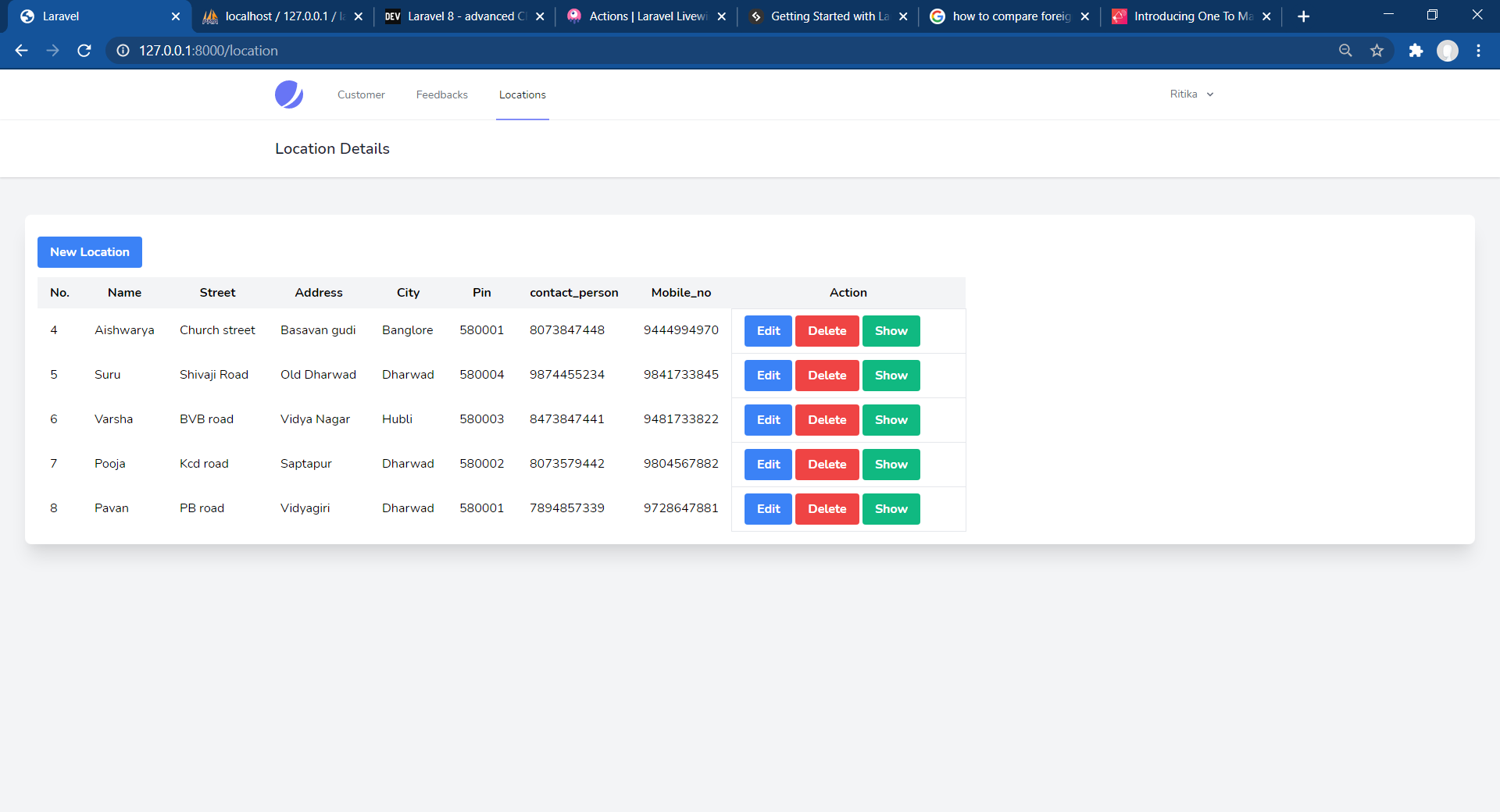
**Customer Page**



**Feedback – Page**



**Location – Page**



**Code – Snippets**

**Schema Code Snippets**

1. Create\_customer\_table

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateCustomersTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('customers', function (Blueprint $table) {

$table->bigincrements('id');

$table->string('name');

$table->date('dob');

$table->string('gender');

$table->string('email');

$table->string('mobile\_no');

$table->text('address');

$table->text('city');

$table->integer('pin');

$table->text('state');

$table->text('country');

$table->biginteger('aadhar\_no');

$table->text('religion');

$table->text('caste');

$table->text('category');

$table->string('emergency\_contacts');

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('customers');

}

}

1. Create\_locations\_table

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateLocationsTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('locations', function (Blueprint $table) {

$table->bigincrements('id');

$table->string('name');

$table->text('street');

$table->text('address');

$table->text('city');

$table->integer('pin');

$table->string('contact\_person');

$table->string('mobile\_no');

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('locations');

}

}

1. Create\_feedbacks\_table

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateFeedbacksTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('feedbacks', function (Blueprint $table) {

$table->bigincrements('id');

$table->text('title');

$table->text('type');

$table->text('description');

$table->text('comments');

$table->text('status');

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('feedbacks');

}

}

**Model Files**

1. Customer.php

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Customer extends Model

{

use HasFactory;

protected $fillable=[

'name',

'dob',

'gender',

'email',

'mobile\_no',

'address',

'city',

'pin',

'state',

'country',

'aadhar\_no',

'religion',

'caste',

'category',

'emergency\_contacts'

];

}

1. Location.php

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Location extends Model

{

use HasFactory;

protected $table='locations';

protected $fillable=[

'name',

'street',

'address',

'city',

'pin',

'contact\_person',

'mobile\_no'

];

}

1. Feedback.php

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Feedback extends Model

{

use HasFactory;

protected $table='feedbacks';

protected $fillable=[

'title',

'type',

'description',

'comments',

'status'

];

}

**Controllers files**

1. CustomerController.php

<?php

namespace App\Http\Livewire;

use Livewire\Component;

use App\Models\Customer;

class Customers extends Component

{

public $customers=[];

public $name, $dob, $gender,$email,$mobile\_no,$address,$city,$pin,$state,$country,$aadhar\_no,$religion,$caste,$category,$emergency\_contacts;

public $isOpen = 0;

public function render()

{

$this->customers = Customer::all();

return view('livewire.customers');

}

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

public function create()

{

$this->resetInputFields();

$this->openModal();

}

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

public function openModal()

{

$this->isOpen = true;

}

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

public function closeModal()

{

$this->isOpen = false;

}

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

private function resetInputFields(){

$this->name = '';

$this->dob = '';

$this->gender = '';

$this->id = '';

$this->email = '';

$this->mobile\_no = '';

$this->address = '';

$this->city = '';

$this->pin = '';

$this->state = '';

$this->country = '';

$this->aadhar\_no = '';

$this->religion = '';

$this->caste = '';

$this->category = '';

$this->emergency\_contacts = '';

}

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

public function store()

{

$this->validate([

'name' => 'required',

'dob' => 'required',

'gender' => 'required',

'email' => 'required',

'mobile\_no' => 'required',

'address' => 'required',

'city' => 'required',

'pin' => 'required',

'state' => 'required',

'country' => 'required',

'aadhar\_no' => 'required',

'religion' => 'required',

'caste' => 'required',

'category' => 'required',

'emergency\_contacts' => 'required',

]);

$data= array(

'name' => $this->name,

'dob' => $this->dob,

'gender' => $this->gender,

'email' => $this->email,

'mobile\_no' => $this->mobile\_no,

'address' => $this->address,

'city' => $this->city,

'pin' => $this->pin,

'state' => $this->state,

'country' => $this->country,

'aadhar\_no' => $this->aadhar\_no,

'religion' => $this->religion,

'caste' => $this->caste,

'category' => $this->category,

'emergency\_contacts' => $this->emergency\_contacts

);

$customer=Customer::updateOrCreate(['id' => $this->id],$data);

session()->flash('message',

$this->id ? 'Customer Updated Successfully.' : 'Customer Created Successfully.');

$this->closeModal();

$this->resetInputFields();

}

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

public function edit($id)

{

$customer = Customer::findOrFail($id);

$this->name = $customer->name;

$this->dob = $customer->dob;

$this->gender = $customer->gender;

$this->id = $customer->id;

$this->email = $customer->email;

$this->mobile\_no = $customer->mobile\_no;

$this->address = $customer->address;

$this->city = $customer->city;

$this->pin = $customer->pin;

$this->state = $customer->state;

$this->country = $customer->country;

$this->aadhar\_no = $customer->aadhar\_no;

$this->religion = $customer->religion;

$this->caste = $customer->caste;

$this->category = $customer->category;

$this->emergency\_contacts = $customer->emergency\_contacts;

$this->openModal();

}

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

public function delete($id)

{

Customer::find($id)->delete();

session()->flash('message', 'Customer Deleted Successfully.');

}

}

1. LocationController.php

<?php

namespace App\Http\Livewire;

use Livewire\Component;

use App\Models\Location;

class Locations extends Component

{

public $locations=[];

public $name, $street, $address,$city,$pin,$contact\_person,$mobile\_no;

public $isOpen = 0;

public function render()

{

$this->locations = Location::all();

return view('livewire.locations');

}

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

public function create()

{

$this->resetInputFields();

$this->openModal();

}

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

public function openModal()

{

$this->isOpen = true;

}

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

public function closeModal()

{

$this->isOpen = false;

}

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

private function resetInputFields(){

$this->name = '';

$this->street = '';

$this->address = '';

$this->city = '';

$this->pin = '';

$this->contact\_person = '';

$this->mobile\_no = '';

}

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

public function store()

{

$this->validate([

'name' => 'required',

'street' => 'required',

'address' => 'required',

'city' => 'required',

'pin' => 'required',

'contact\_person' => 'required',

'mobile\_no' => 'required',

]);

Location::updateOrCreate(['id' => $this->id], [

'name' => $this->name,

'street' => $this->street,

'address' => $this->address,

'city' => $this->city,

'pin' => $this->pin,

'contact\_person' => $this->contact\_person,

'mobile\_no' => $this->mobile\_no

]);

session()->flash('message',

$this->id ? 'Location Updated Successfully.' : 'Location Created Successfully.');

$this->closeModal();

$this->resetInputFields();

}

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

public function edit($id)

{

$location = Location::findOrFail($id);

$this->name = $location->name;

$this->street = $location->street;

$this->address = $location->address;

$this->city = $location->city;

$this->pin = $location->pin;

$this->contact\_person = $location->contact\_person;

$this->mobile\_no = $location->mobile\_no;

$this->openModal();

}

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

public function delete($id)

{

Location::find($id)->delete();

session()->flash('message', 'Location Deleted Successfully.');

}

}

1. FeedbackController.php

<?php

namespace App\Http\Livewire;

use Livewire\Component;

use App\Models\Feedback;

class Feedbacks extends Component

{

public $feedbacks=[];

public $title, $type, $description,$comments,$status;

public $isOpen = 0;

public function render()

{

$this->feedbacks = Feedback::all();

return view('livewire.feedbacks');

}

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

public function create()

{

$this->resetInputFields();

$this->openModal();

}

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

public function openModal()

{

$this->isOpen = true;

}

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

public function closeModal()

{

$this->isOpen = false;

}

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

private function resetInputFields(){

$this->title = '';

$this->type = '';

$this->description = '';

$this->comments = '';

$this->status = '';

}

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

public function store()

{

$this->validate([

'title' => 'required',

'type' => 'required',

'description' => 'required',

'comments' => 'required',

'status' => 'required',

]);

Feedback::updateOrCreate(['id' => $this->id], [

'title' => $this->title,

'type' => $this->type,

'description' => $this->description,

'comments' => $this->comments,

'status' => $this->status,

]);

session()->flash('message',

$this->id ? 'Feedback Updated Successfully.' : 'Feedback Created Successfully.');

$this->closeModal();

$this->resetInputFields();

}

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

public function edit($id)

{

$feedback = Feedback::findOrFail($id);

$this->title = $customer->title;

$this->type = $customer->type;

$this->description = $customer->description;

$this->comments = $customer->comments;

$this->status = $customer->status;

$this->openModal();

}

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

public function delete($id)

{

Feedback::find($id)->delete();

session()->flash('message', 'Feedback Deleted Successfully.');

}

}

**Blade Files**

1. Customers.blade.php

<x-slot name="header">

<h2 class="font-semibold text-xl text-gray-800 leading-tight">

Customer Details

</h2>

</x-slot>

<div class="py-12">

<div class="max-w-full mx-auto lg:px-8" style="width: 100%;">

<div class="bg-white overflow-hidden shadow-xl sm:rounded-lg px-4 py-4">

@if (session()->has('message'))

<div class="bg-teal-100 border-t-4 border-teal-500 rounded-b text-teal-900 px-4 py-3 shadow-md my-3" role="alert">

<div class="flex">

<div>

<p class="text-sm">{{ session('message') }}</p>

</div>

</div>

</div>

@endif

<button wire:click="create()" class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded my-3"> New Customer</button>

@if($isOpen)

@include('livewire.create')

@endif

<table class="table table-bordered table-responsive-lg" >

<thead>

<tr class="bg-gray-100">

<th class="px-4 py-2">No.</th>

<th class="px-4 py-2">Name</th>

<th class="px-4 py-2">Dob</th>

<th class="px-4 py-2">Gender</th>

<th class="px-4 py-2" >Email</th>

<th class="px-4 py-2">Mobile\_no</th>

<th class="px-4 py-2">Address</th>

<th class="px-4 py-2">City</th>

<th class="px-4 py-2">Pin</th>

<th class="px-4 py-2">State</th>

<th class="px-4 py-2">Country</th>

<th class="px-4 py-2">Aadhar\_no</th>

<th class="px-4 py-2">Religion</th>

<th class="px-4 py-2">Caste</th>

<th class="px-4 py-2">Category</th>

<th class="px-4 py-2">Emergency\_contacts</th>

<th width="300px">Action</th>

</tr>

</thead>

<tbody>

@foreach ($customers as $customer)

<tr>

<td class="px-4 py-2">{{ $customer->id }}</td>

<td class="px-4 py-2">{{ $customer->name }}</td>

<td class="px-4 py-2">{{ $customer->dob }}</td>

<td class="px-4 py-2">{{ $customer->gender }}</td>

<td class="px-4 py-2">{{ $customer->email }}</td>

<td class="px-4 py-2">{{ $customer->mobile\_no }}</td>

<td class="px-4 py-2">{{ $customer->address }}</td>

<td class="px-4 py-2">{{ $customer->city }}</td>

<td class="px-4 py-2">{{ $customer->pin }}</td>

<td class="px-4 py-2">{{ $customer->state }}</td>

<td class="px-4 py-2">{{ $customer->country }}</td>

<td class="px-4 py-2">{{ $customer->aadhar\_no }}</td>

<td class="px-4 py-2">{{ $customer->religion }}</td>

<td class="px-4 py-2">{{ $customer->caste }}</td>

<td class="px-4 py-2">{{ $customer->category }}</td>

<td class="px-4 py-2">{{ $customer->emergency\_contacts }}</td>

<td class="border px-4 py-2">

<button wire:click="edit({{ $customer->id }})" class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded">Edit</button>

<button wire:click="delete({{ $customer->id }})" class="bg-red-500 hover:bg-red-700 text-white font-bold py-2 px-4 rounded">Delete</button>

<button wire:click="show({{'livewire.show',$customer->id}})" class="bg-green-500 hover:bg-green-700 text-white font-bold py-2 px-4 rounded">Show</button>

</td>

</tr>

@endforeach

</tbody>

</table>

</div>

</div>

</div>

1. Locations.blade.php

<x-slot name="header">

<h2 class="font-semibold text-xl text-gray-800 leading-tight">

Location Details

</h2>

</x-slot>

<div class="py-12">

<div class="max-w-full mx-auto lg:px-8" style="width: 100%;">

<div class="bg-white overflow-hidden shadow-xl sm:rounded-lg px-4 py-4">

@if (session()->has('message'))

<div class="bg-teal-100 border-t-4 border-teal-500 rounded-b text-teal-900 px-4 py-3 shadow-md my-3" role="alert">

<div class="flex">

<div>

<p class="text-sm">{{ session('message') }}</p>

</div>

</div>

</div>

@endif

<button wire:click="create()" class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded my-3"> New Location</button>

@if($isOpen)

@include('livewire.createlocation')

@endif

<table class="table table-bordered table-responsive-lg" >

<thead>

<tr class="bg-gray-100">

<th class="px-4 py-2">No.</th>

<th class="px-4 py-2">Name</th>

<th class="px-4 py-2">Street</th>

<th class="px-4 py-2">Address</th>

<th class="px-4 py-2" >City</th>

<th class="px-4 py-2">Pin</th>

<th class="px-4 py-2">contact\_person</th>

<th class="px-4 py-2">Mobile\_no</th>

<th width="300px">Action</th>

</tr>

</thead>

<tbody>

@foreach ($locations as $location)

<tr>

<td class="px-4 py-2">{{ $location->id }}</td>

<td class="px-4 py-2">{{ $location->name }}</td>

<td class="px-4 py-2">{{ $location->street }}</td>

<td class="px-4 py-2">{{ $location->address }}</td>

<td class="px-4 py-2">{{ $location->city }}</td>

<td class="px-4 py-2">{{ $location->pin }}</td>

<td class="px-4 py-2">{{ $location->contact\_person }}</td>

<td class="px-4 py-2">{{ $location->mobile\_no }}</td>

<td class="border px-4 py-2">

<button wire:click="edit({{ $location->id }})" class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded">Edit</button>

<button wire:click="delete({{ $location->id }})" class="bg-red-500 hover:bg-red-700 text-white font-bold py-2 px-4 rounded">Delete</button>

<button wire:click="show()" class="bg-green-500 hover:bg-green-700 text-white font-bold py-2 px-4 rounded">Show</button>

</td>

</tr>

@endforeach

</tbody>

</table>

</div>

</div>

</div>

1. Feedback.blade.php

<x-slot name="header">

<h2 class="font-semibold text-xl text-gray-800 leading-tight">

Feedback Details

</h2>

</x-slot>

<div class="py-12">

<div class="max-w-full mx-auto lg:px-8" style="width: 100%;">

<div class="bg-white overflow-hidden shadow-xl sm:rounded-lg px-4 py-4">

@if (session()->has('message'))

<div class="bg-teal-100 border-t-4 border-teal-500 rounded-b text-teal-900 px-4 py-3 shadow-md my-3" role="alert">

<div class="flex">

<div>

<p class="text-sm">{{ session('message') }}</p>

</div>

</div>

</div>

@endif

<button wire:click="create()" class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded my-3"> New Feedback</button>

@if($isOpen)

@include('livewire.createfeedback')

@endif

<table class="table table-bordered table-responsive-lg" >

<thead>

<tr class="bg-gray-100">

<th class="px-4 py-2">No.</th>

<th class="px-4 py-2">Title</th>

<th class="px-4 py-2">Type</th>

<th class="px-4 py-2">Description</th>

<th class="px-4 py-2" >Comments</th>

<th class="px-4 py-2">Status</th>

<th width="300px">Action</th>

</tr>

</thead>

<tbody>

@foreach ($feedbacks as $feedback)

<tr>

<td class="px-4 py-2">{{ $feedback->id }}</td>

<td class="px-4 py-2">{{ $feedback->title }}</td>

<td class="px-4 py-2">{{ $feedback->type }}</td>

<td class="px-4 py-2">{{ $feedback->description }}</td>

<td class="px-4 py-2">{{ $feedback->comments }}</td>

<td class="px-4 py-2">{{ $feedback->status }}</td>

<td class="border px-4 py-2">

<button wire:click="edit({{ $feedback->id }})" class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded">Edit</button>

<button wire:click="delete({{ $feedback->id }})" class="bg-red-500 hover:bg-red-700 text-white font-bold py-2 px-4 rounded">Delete</button>

<button wire:click="show()" class="bg-green-500 hover:bg-green-700 text-white font-bold py-2 px-4 rounded">Show</button>

</td>

</tr>

@endforeach

</tbody>

</table>

</div>

</div>

</div>

**7. Conclusion and Future scope**

Customer relationship management is focused on the creation and maintenance of

long-term, mutually beneficial relationships with strategically important markets.

Customer relationship management is focused on the creation and maintenance of

long-term, mutually beneficial relationships with strategically important markets.

Customer relationship management is focused on the creation and maintenance of

long-term, mutually beneficial relationships with strategically important markets.

Customer relationship management is focused on the creation and maintenance of

long-term, mutually beneficial relationships with strategically important markets. It is

based on the premise that customers with the highest life-time value potential are

those in whom the company should invest their retention resources. Other customers

might be fired. For others, it may be possible to re-engineer or nurture the

relationship to create new sources of value.

Customer relationship management is focused on the creation and maintenance of

long-term, mutually beneficial relationships with strategically important markets

Customer relationship management is focused on the creation and maintainence of long-term, mutually beneficial relationships with strategically important markets.

Creating, communicating and delivering value to selected customers can only be achieved if the company aligns and co-ordinates its relationships with four other major constituencies: suppliers, owners/investors, employees and partners. Together, these five constituencies form the S.C.O.P.E. of customer relationship management. Another fundamental of customer relationship management is that the value proposition or offer should be customized to meet or exceed customer expectations. Traditionally, customization has centered on the product component of the value proposition. Equally, it is possible to customize service, process, people, distribution, price and communication.

**8.Reference/Bibliography**

* <https://dev.to/kingsconsult/laravel-8-auth-registration-and-login-32jl>
* <https://www.positronx.io/laravel-pdf-tutorial-generate-pdf-with-dompdf-in-laravel/>
* <https://websolutionstuff.com/post/laravel-8-pdf-generate-example>
* <https://www.itsolutionstuff.com/post/laravel-livewire-crud-application-tutorialexample.html>