

SMART CLEAN INDIA

A

Major Project Report

Submitted by:

AKASH SONI (0887CS161002)

LAKSHIT PATIDAR (0887CS161016)

GAJENDRA DAWAR (0887CS161010)

VIKRAM BAMANIYA (0887CS173D02)

BACHELOR OF ENGINEERING IN COMPUTER SCIENCE & ENGINEERING

Under the Guidance

of

Mr. Anurag Kumar

(Asst.Professor, CSE)



DR. APJ ABDUL KALAM UNIVERSITY INSTITUTE OF TECHNOLOGY

JHABUA, M.P. (INDIA) - 457661

(AFFILIATED TO RGPV, BHOPAL, M.P. (INDIA))

JUNE -2020

DECLARATION

We AKASH SONI (0887CS161002), LAKSHIT PATIDAR (0887CS161016), GAJENDRA DAWAR (0887CS161010), VIKRAM BAMANIYA (0887CS173d02), hereby declare that this project work entitled “SMART CLEAN INDIA” was carried out by us under the supervision of Prof. Anurag Kumar, Department of CSE, Dr. APJ Abdul Kalam University Institute Of Technology, Jhabua (M.P.). This project work is submitted to Department of Computer Science and Engineering during the academic year 2019-2020.

Place:

Date:

Name

Signature

AKASH SONI (0887CS161002)
LAKSHIT PATIDAR (0887CS161016)
GAJENDRA DAWAR (0887CS161010)
VIKRAM BAMANIYA (0887CS173D02)

DR. APJ ABDUL KALAM UNIVERSITY INSTITUTE OF TECHNOLOGY, JHABUA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



SESSION 2019-2020

CERTIFICATE OF APPROVAL

This is to certify that the work embodies in this report entitled {“ **SMART CLEAN INDIA**”} being submitted by **AKASH SONI (0887CS161002) & LAKSHIT PATIDAR (0887CS161016), GAJENDRA DAWAR (0887CS161010), VIKRAM BAMANIYA (0887CS173D02)** who carried out the project work under our supervision and guidance in the “**Department of Computer Science & Engineering**”, **Dr. APJ Abdul Kalam University Institute Of Technology, Jhabua (M.P.)**.

(Internal Examiner)

(External Examiner)

Name:

Name:

Designation: AP

Designation:

Institute: Dr. APJ Abdul Kalam UIT, Jhabua

Institute:

Date:

Date:

DR. APJ ABDUL KALAM UNIVERSITY INSTITUTE OF TECHNOLOGY, JHABUA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



SESSION 2019-2020

CERTIFICATE

This is to certify that the work embodies in this report entitled “**SMART CLEAN INDIA**” being submitted by **AKASH SONI (0887CS161002)**, **LAKSHIT PATIDAR (0887CS161016)**, **GAJENDRA DAWAR (0887CS161010)**, **VIKRAM BAMANIYA (0887CS173D02)**, who carried out the project work under my supervision and guidance in the “**Department of Computer Science & Engineering**”, Dr. APJ Abdul Kalam UIT, Jhabua (M.P.).

Guided & Approved By:

Forwarded By:

Prof. Anurag Kumar

Asst. prof.
Department of CSE
Dr. APJ Abdul Kalam UIT, Jhabua

Prof. (Mrs.) Vaishali Ahirwar

HOD
Department of CSE
Dr. APJ Abdul Kalam UIT, Jhabua

ABSTRACT

An Engineer is originally a profession who can solve the problem with optimism.

Our major is based to solve a day to day problem of most of the population. We had a core aim to make India clean and hygienic.

As we know, it is the fundamental duty of every citizen of India to take care to his motherland but a common people is busy in his daily stuffs and can't contribute a lot for his nation as a socialist.

We have developed a solution for collecting garbage from the city on the populace request. Our platform will provide a source to those who register on our website and just raise a request for garbage collection which is gathered in their livelihood.

Our website is connected with the government authorities which are responsible for garbage collection with the city/District. Our user only have to register on our website and register a request for garbage with the picture of garbage and location.

After the registration of request, the people of that area will get notified about that request registered by that user to cross verify. The people of that area can verify the request if it had genuineness otherwise it will be marked as fake by the people.

After the verification of the request, it has been sent to the Nagar Palika/Nagar Nigam the respective governing authority to take action for that request.

So simply a request by just registering on fingertips can solve the problem of garbage in your locality.

Now people don't have to go government offices for garbage collection stuffs just use our platform and get rid off your nearby garbage.

ACKNOWLEDGEMENT

Knowledge is expression of experience gained in life. It is the choicest possession that should be happily shared with others.

In this regards I/We feel great pleasure in submitting this minor project report on “SMART CLEAN INDIA”. It gives me immense pleasure to express my deepest sense of gratitude and sincere thanks to my highly respected and esteemed guide “Prof. Anurag Kumar” (Dept. of CSE), Dr. APJ Abdul Kalam UIT, Jhabua for their valuable guidance, encouragement and help for completing this work. Their useful suggestions for this whole work and co-operative behavior are sincerely acknowledged. During this project, we received a lot of help, advice and cooperation from our esteemed faculty and other distinguished persons. We would also like to thank “Prof. (Mrs.) Vaishali Ahirwar” (H.O.D. Dept. of CSE) for their valuable guidance through the course of project without whose encouragement the project wouldn’t have been a success.

We are grateful to our Principal Dr. “K.S. Chandel” and college authorities for their support and all those who have directly or indirectly helped us during the project work.

At the end I would like to express my sincere thanks to all my friends and others who helped me directly or indirectly during this project work.

Submitted by

AKASH SONI (0887CS161002)

LAKSHIT PATIDAR (0887CS161016)

GAJENDRA DAWAR (0887CS161010)

VIKRAM BAMANIYA (0887CS173D02)

TABLE OF CONTENTS

Declaration	i
Certificate of approval	ii
Certificate	iii
Acknowledgement	iv
Abstract	v
List of Figures	vi
List of Tables	vii
 1.Introduction	
1.1 Problem Definition	10
1.2 Aim and Objective	11
1.3 Project Specification	12
2.Analysis	
2.1 System Requirement analysis	13
2.2 System Feasibility	14
2.2.3 Operational Feasibility	
2.2.2 Technical feasibility	
2.2.1 Economic Feasibility	
2.3 Functional Requirements	15
2.4 Non- Functional Requirements	16
2.4.1 Safety Requirements	
2.4.2 Security Requirements	
2.4.3 Software Quality Attributes	
2.5.1 Hardware Requirements	
2.5 Platform Specification	17
2.5.1 Hardware Requirements	

2.5.2 Software Requirements	
2.6 User Case Diagram	18
3.Methodology	19
4.Design	
4.1. Architecture	24
4.1.1 What is MVC?	
4.1.2 Why use MVC?	
4.2 E-R Diagram	27
4.3 Activity Diagram	28
4.4 Data flow Diagram	29
4.4.1 Context level data flow diagram	
4.4.2 Data flow diagram for Administrator	
4.5 Database Design	31
4.5.1 Database Details	
5. .Implementation	
5.1 MVC implementation in Laravel	33
5.2 User Functionality	43
5.3 Government authorities Functionality	51
5.4 Administrator Functionality	53
Future Enhancements	
Conclusion	
Appendices	
References	

List of Figures

Figure 1. User Case Diagram	18
Figure 2. Waterfall model	20
Figure-3 Expansion of the waterfall model and the application of MVC Architecture.	22
Figure 4. MVC Architecture	25
Figure 5. Entity Relationship Diagram(ERD)	27
Figure 6. Activity Diagram of User	28
Figure 7. Context level Data flow Diagram	29
Figure 8. Data flow Diagram for Administrator	30
Figure 9. Class Diagram	31

1. INTRODUCTION

1.1 Problem Definition

Our project is a initiative under “The Swachh Bharat Abhiyan”. The Swachh Bharat Abhiyan is the most significant cleanliness campaign by the Government of India.

Cleanliness refers to the state of being clean. It is something which must not be forced but encouraged. Cleanliness is a good habit which can enhance the quality of one’s life.

Similar to the basic essentials of life like food, water, shelter, cleanliness also holds great significance in life. It is, in fact, one of the most important things for healthy living. The first and foremost importance of cleanliness is that it means the absence of disease. Cleanliness helps us stay refreshed and hygienic on a personal level.

Our project deals with the cleanliness at a society level

Our project is implemented in laravel(a framework of php). For designing we use bootstrap ,Html ,Css, Javascript etc.

1.2 Aim and Objective

We have developed a website for collecting the garbage from the location by the responsible authorities of government.

Our project act as a medium for the common people to convey their complaint of garbage gathering to the government officials without visiting to the government offices or stuck for hours in these offices.

We offer the solution at your fingertips just register on our website and register a request.

After the registration of request, the people of that area will get notified about that request registered by that user to cross verify. The people of that area can verify the request if it had genuineness otherwise it will be marked as fake by the people.

After the verification of the request, it has been send to the nagar palika/nagar nigam the respective governing authority to take action for that request.

So simply register a request on fingertips can solve the problem of garbage in your locality.

1.3 Project Specifications

Brief about Laravel

Laravel is a web application framework with expressive, elegant syntax. We believe development must be an enjoyable, creative experience to be truly fulfilling. Laravel attempts to take the pain out of development by easing common tasks used in the majority of web projects, such as authentication, routing, sessions, and caching.

Laravel aims to make the development process a pleasing one for the developer without sacrificing application functionality. Happy developers make the best code. To this end, we've attempted to combine the very best of what we have seen in other web frameworks, including frameworks implemented in other languages, such as Ruby on Rails, ASP.NET MVC, and Sinatra.

Laravel is accessible, yet powerful, providing powerful tools needed for large, robust applications. A superb inversion of control container, expressive migration system, and tightly integrated unit testing support give you the tools you need to build any application with which you are tasked.

2 .ANALYSIS

2.1 System requirement analysis

We are using MySQL or Redis or anything else then you should be aware of that in your application and know what you need. You'd also need a web server more than likely, Nginx or Apache2 are supported by Laravel out-of-the-box. You can use whatever operating system you want that fits these other software requirements, often people choose a Linux distribution such as Ubuntu.

Hardware details are another concern entirely, that depends on what kind of traffic you're expecting, what kind of processing your application has to do, etc. It's very different if you have to encode 2GB video files from millions of users compared to a simple web page with static HTML and a form.

2.2 System Feasibility

The prime focus of the feasibility is evaluating the practicality of the proposed system keeping in mind a number of factors. The following factors are taken into account before deciding in favor of the new system.

2.2.1 Economic Feasibility

Currently, user have to go to Nagar palika offices to register a complaint and have to expend a lot of time in offices. therefore by our website time will be saved, resources like fuel will be saved because of not going to any office now.

2.2.2 Technical feasibility

Keeping in view the above fact, nowadays all organizations are automating the repetitive and monotonous works done by humans. The key process areas of the current system are nicely amenable to automation and hence the technical feasibility is proved beyond doubt.

2.2.3 Operational Feasibility

The present system has automated most of the manual tasks. Therefore the proposed system will increase the operational efficiency of the administrator and instructors.

2.3 Functional Requirements

- User should have to fill the genuine information in the registration form because we don't have any resource to verify that the user information is original or fake.
- User have to response in time for request verification either there will be a long delay to take action by the officials due to not confirmation of the request.
- Secure registration and profile management facilities for different users.
- For request verification it will show a complete detail of the location where garbage is gathered so user also have to emphasize about the given address along with the genuinity.
- There is no notification system for the user which can notify user about the new upcoming request so the user have to regularly check the request for verification section although to verify the request and further actions can be taken.

2.4 Non- Functional Requirements

2.4.1 Safety Requirements

If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed up log, up to the time of failure.

2.4.2 Security Requirements

Security systems need database storage just like many other applications. However, the special requirements of the security market mean that vendors must choose their database partner carefully.

2.4.3 Software Quality Attributes

- **AVAILABILITY:** Since we are hosting our project on the local server it will be available all the time.
- **CORRECTNESS:** The system should generate an appropriate report about different activities of the requests and should keep track of all records.
- **MAINTAINABILITY:** The system should maintain documentation of all the request registrations.
- **USABILITY:** The system should satisfy the maximum number of users needs.

2.5 Platform Specification

2.5.1 Hardware Requirements

- Pentium IV or higher, (PIV-300GHz recommended)
- 256 MB RAM
- 1 Gb hard free drive space

2.5.2 Software Requirements

The Laravel framework has a few system requirements. Of course, all of these requirements are satisfied by the Laravel Homestead virtual machine, so it's highly recommended that you use Homestead as your local Laravel development environment.

However, if you are not using Homestead, you will need to make sure your server meets the following requirements:

- PHP \geq 5.6.4
- OpenSSL PHP Extension
- PDO PHP Extension
- Mbstring PHP Extension
- Tokenizer PHP Extension
- XML PHP Extension

Above listed are the requirements for laravel however there will be a mid-level configuration of the system for smooth execution of the website.

2.6 User Case Diagram

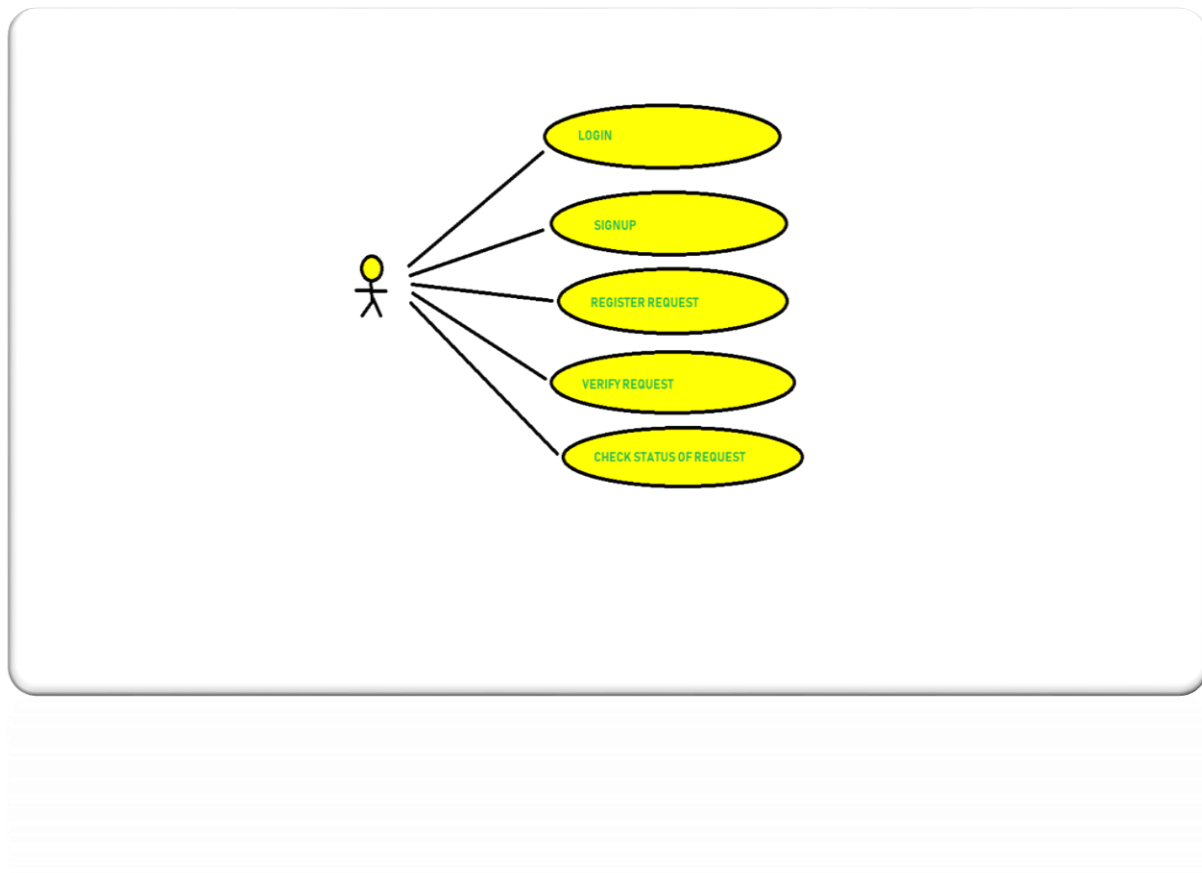


Fig 1- User case Diagram

3.Methodology

Web application development is always growing rapidly through the years. It requires the system to work quickly and accurately. Many developers are trying to create new and different technologies, especially for building faster and more powerful web applications. Until now, web application like front-end or client-side still using HTML, CSS, and JavaScript to present data. While the back-end or server-side refers to the storage and processing of data that normally handled by PHP.

Today, information technology is one of the fastest growing businesses areas. The software contains three major modules, namely the interface, business logic, and data All three are closely related so that software developers have to write all the code for production applications.

Currently, with the MVC design, developers do not have to do that because it serves to separate the input, business logic, and output and it also makes model classes reusable without modification SDLC model consists of a sequence of steps that must be followed by system developers.

Therefore, many companies engaged in software development chose this model to cause it is easy to make applications with high quality. One of the SDLC models, namely the Waterfall, which consists of five subsequent stages, there are business analysis, design, implementation, testing, and maintenance. The success of the Waterfall model makes many software development companies adopting the main framework for the design, build, and maintain an application .Here the concept of MVC for software development using SDLC techniques, especially the expansion of the Waterfall model The goal is to optimize the Waterfall models with

MVC architecture, so it will get higher quality applications regarding safety and minimises preprocessing time.

Waterfall Method

In 1970, one of the articles Walter Royce is the first publication that discusses the waterfall model. Older models widely criticized and modified so as to produce new models and different. The waterfall model of designing things related to software development before starting coding and Successfully build the system. The main function of the waterfall model is to ensure the success of the project Waterfall method consists of several phases of the software development process and there are requirement analysis, design, implementation testing, and maintenance.

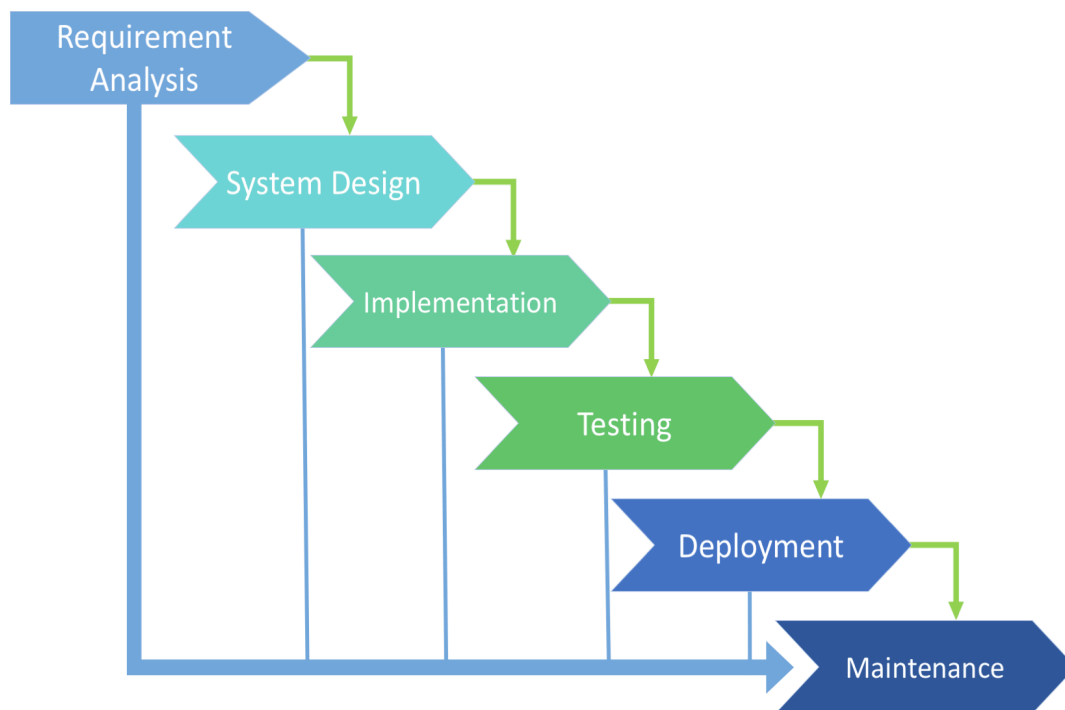


Fig 2– Waterfall Model

1. Requirements Analysis This stage called Software Required Specification (SRS). The main focus at this stage is a complete description of the software to be developed. The project manager should discuss with users of the system to analyze the functional and nonfunctional requirements for the fulfillment of this stage Functional requirements typically include the purpose, scope,

perspective, function, we characteristics, interface, and database. Whereas the nonfunctional requirement is referring to criteria,constraint, limitation, and software performance.

2. Design At this stage not only about interface design, but also algorithm design, software architectural design, database, concept design, and structural design. System and software design are made of the requirement specification that is learned in the first phase. The system design help in determining hardware and defining the overall system of architecture.

3. Implementation This stage is the realization of business requirements and system design into the application program Database starts with coding and deploys it. At this stage, the real code was written and compiled info an operational application, where the database and other needs have been learned in the previous stage. The work is divided in modular form to facilitate the development as teamwork. The focus at this stage is the development of software so that this stage is the longest stage of the SDLC.

4. Testing It is also known as the verification and validation that is a process to check that the software meets the original requirements and complete the objectives that have been set. In fact, verification is a software evaluation process to determine whether a product of a given development phase satisfy the conditions imposed at the beginning of that stage. Meanwhile, validation is the process of evaluating software during or at the end of the development process to determine whether it fulfill the requirements specified or not. Also, the testing phase is an outlet for debugging where bugs and system disorders discovered, corrected and improved

5. Maintenance It is the process of modifying the software after it was created and deployed, correcting errors, and improve performance and quality. Additional maintenance activities can be done in this phase includes software adaptation to its environment, to accommodate the needs of new users and improve software reliability.

The main focus of this research is on the expansion of the waterfall model. The waterfall phase, is divided into five stages But the concept that we propose is the expansion of the waterfall model remain in 5 stages but has a different groove. Also, there is a stage in MVC architecture development which is implementation stage shows how the lane of the waterfall model with the expansion of the MVC architecture.

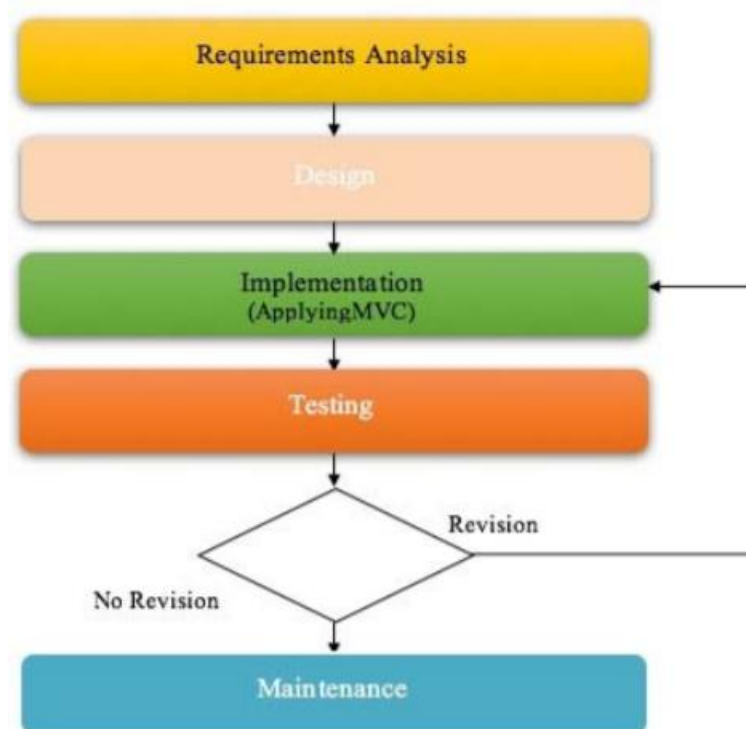


Fig-3 Expansion of the waterfall model and the application of MVC Architecture.

The expansion lane of the waterfall model starts from the analysis of needs. After analysis of needs is assessed, then the data will be designed from the user interface, application logic, database, and so on. Later in the implementation phase with the software is written through coding In its development, we apply the MVC architecture at this stage. Because at this stage of development, there definitely be many obstacles around the business logic, interface, and

database. So with the MVC architecture, the problem will be solved. After the application is completed, then we can do a test. If the applications were built without any flaws, then the system can be directly to the maintenance phase without revised.

However, if the system is still uncompleted, then the system will return to the stage of implementation and do the revised. Once the system is back to be tested to maximize the application functionality

Implementation stage is the stage of making the information system according to the results at the design stage. We build a simple application and its called "Microteaching" to test this model.

Implementation of this system is using the framework laravel. Framework laravel facilitate the implementation of the design stage into the system because it uses MVC architecture. The case of using framework Laravel is the separation of program code based on the Model-View-Controller pattern, implementing the functional requirements into separate modules, and the availability of build in library so as to accelerate the implementation process.

Conclusions: Based on the proposals above, it says that the expansion of the waterfall model is used by many large companies for their internal projects. Due in large companies most clients can change their requirements at any specified time. Not only time, usually the client wants to change the graphical display totally at any time. So it does not waste time and would hold up the process of software

However, with the application of MVC architecture in software development, application developers can easily change the parts that need to be changed suddenly. Because this framework separates View, Controllers and Model. Therefore, this framework is very stable, efficient and able to create high-quality applications.

4.DESIGN

4.1. Architecture

Laravel framework implements MVC architecture. MVC is a software architecture pattern and it stands for Model View Controller.

4.1.1 What is MVC?

MVC is an acronym for ‘Model View Controller’. It represents architecture developers adopt when building applications. With the MVC architecture, we look at the application structure with regards to how the data flow of our application works

MVC is a software architecture...that separates domain/application/business...logic from the rest of the user interface. It does this by separating the application into three parts: the model, the view, and the controller.

The model manages fundamental behaviors and data of the application. It can respond to requests for information, respond to instructions to change the state of its information, and even notify observers in event-driven systems when information changes. This could be a database or any number of data structures or storage systems. In short, it is the data and data-management of the application.

The view effectively provides the user interface element of the application. It'll render data from the model into a form that is suitable for the user interface. The controller receives user input and makes calls to model objects and the view to perform appropriate actions.

All in all, these three components work together to create the three basic components of MVC.

– [Bob, Stack Overflow](#)

We have a structure that looks like this:

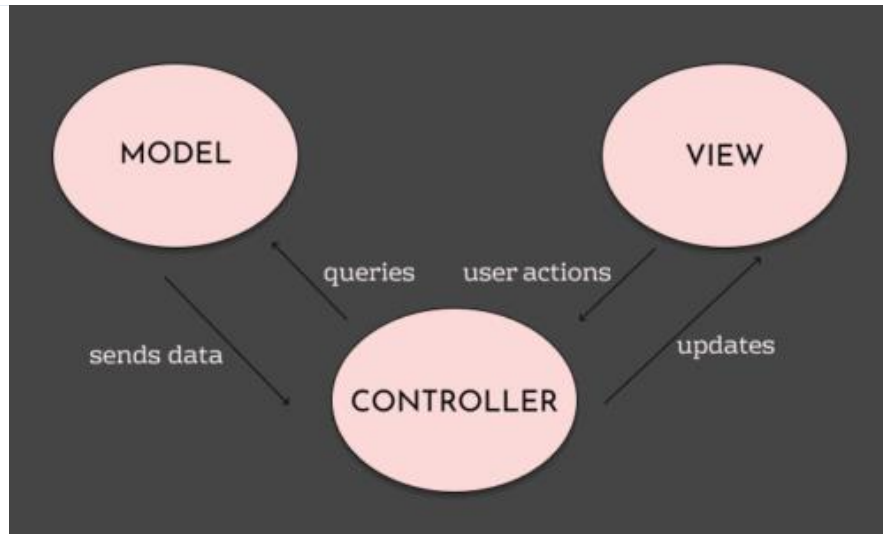


Fig-4 – MVC Architecture

A **Model** is a representation of a real-life instance or object in our code base. The **View** represents the interface through which the user interacts with our application. When a user takes an action, the **Controller** handles the action and updates the **Model** if necessary.

Let's look at a simple scenario.

If you go to an e-commerce website, the different pages you see are provided by the **View** layer. When you click on a particular product to view more, the **Controller** layer processes the user's action. This may involve getting data from a data source using the **Model** layer. The data is then bundled up together and arranged in a **View** layer and displayed to the user. Rinse and repeat.

4.1.2 Why use MVC?

This article is not a comparison between architecture types but information on a single type, which is MVC.

When building PHP applications, it may be okay to have a lot of files flying around in very very small projects. However, when the project becomes even slightly bigger than five files or entry points having a structure can drastically improve maintainability.

When you have to work with codebases that have no architecture, it will become extremely grueling, especially if the project is big and you have to deal with unstructured code laying everywhere. Using MVC can give your code some structure and make it easier to work with.

On a more technical note, when you build using the MVC architecture, you have the following strategic advantages:

- **Splitting roles in your project are easier.**

When the MVC architecture is adopted, you have the advantage of splitting roles in the project. You can have a backend developer working on the controller logic, while a frontend developer works on the views. **Structurally 'a-okay'.**

MVC can force you to split your files into logical directories which makes it easier to find files when working on large projects.

- **Responsibility isolation.**

When you adopt MVC, each broad responsibility is isolated. For instance, you can make changes in the views and the models separately because the model does not depend on the views.

- **Full control of application URLs.**

With MVC architecture, you have full control over how your application appears to the world by choosing the application routes. This comes in handy when you are trying to improve your application for SEO purposes.

- **Writing SOLID code is easier.**

With MVC it is easier to follow the **SOLID** principle.

4.2. E-R Diagram

Entity Relationship Diagram

Below given is a Entity Relationship diagram of our project.

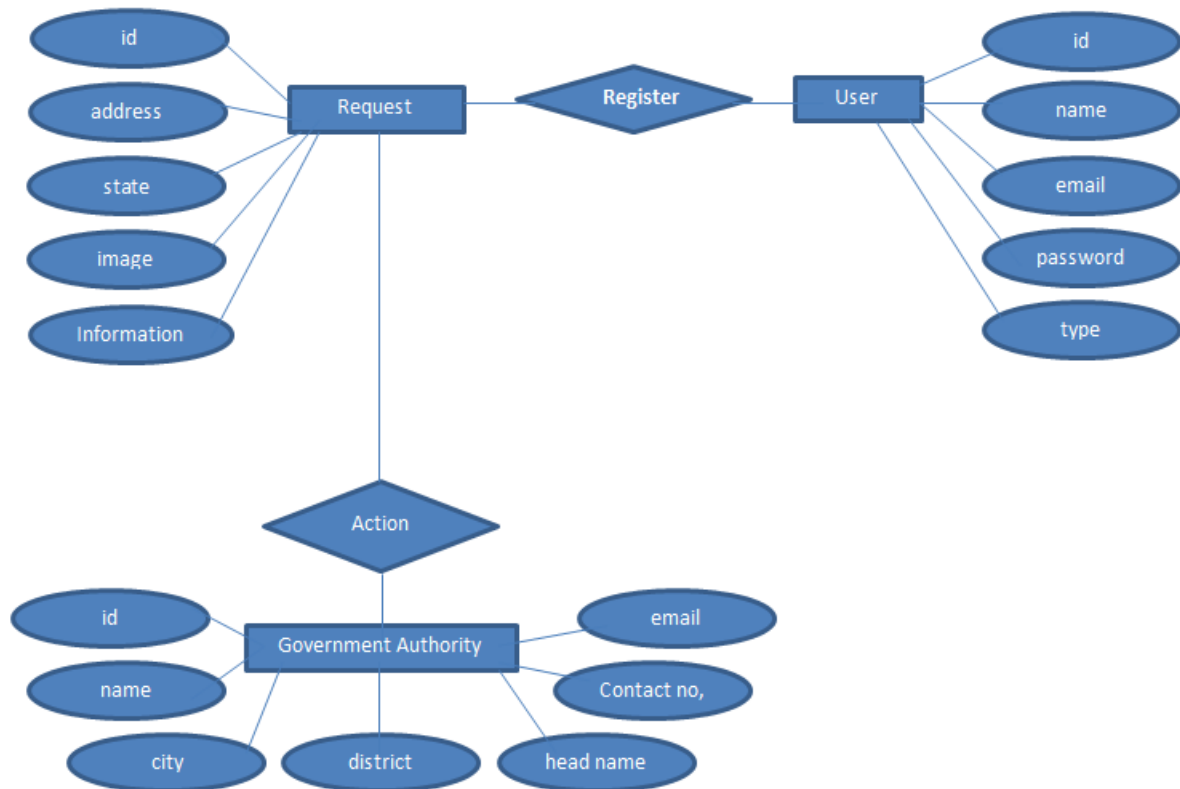


Fig 5 - ER Diagram

This Diagram shows all the relationship between all the entities of the project.

4.3. Activity Diagram

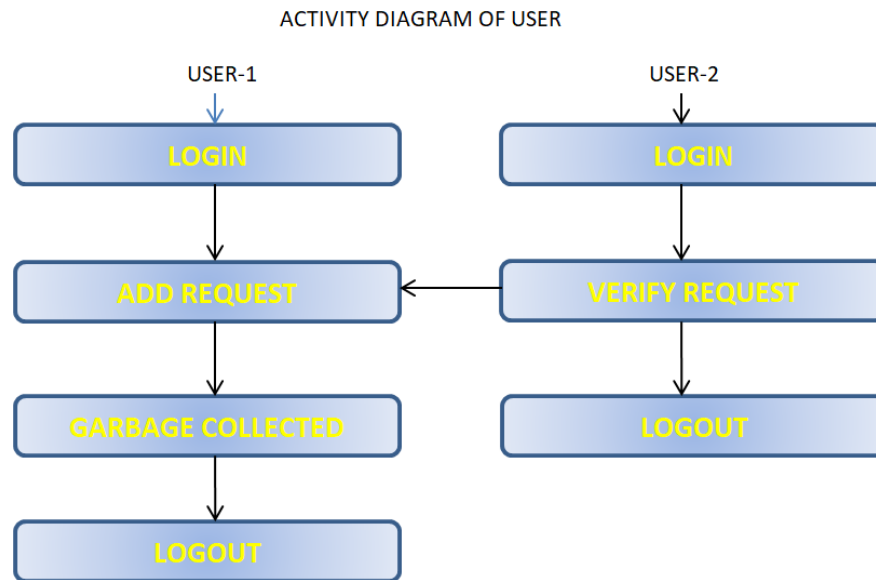


Fig 6– Activity Diagram

4.4. Data flow Diagram

4.4.1 Context level Data flow Diagram

CONTEXT LEVEL DATA FLOW DIAGRAM

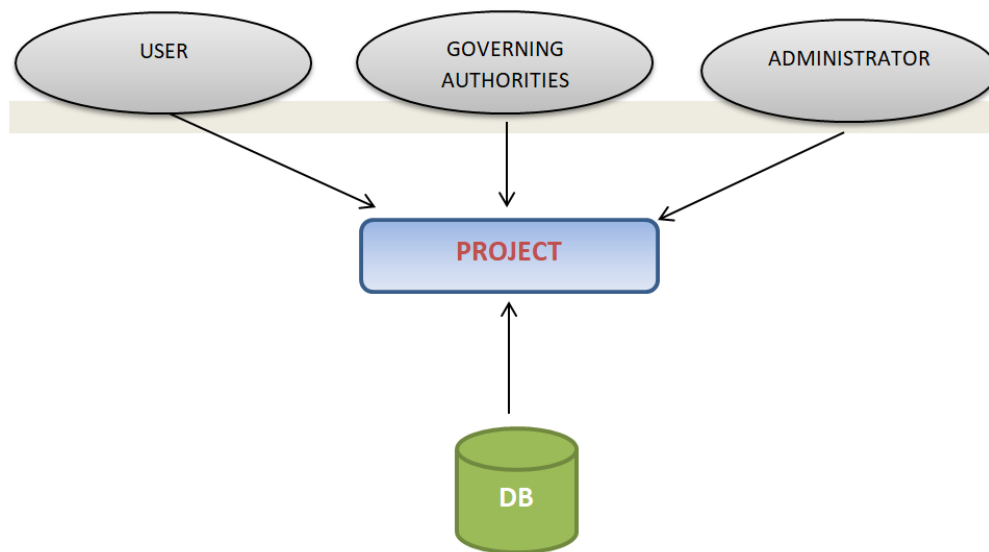


Fig 7- Context level data flow diagram

4.4.2. Data Flow Diagram for Administrator

DATA FLOW DIAGRAM FOR ADMINISTRATOR

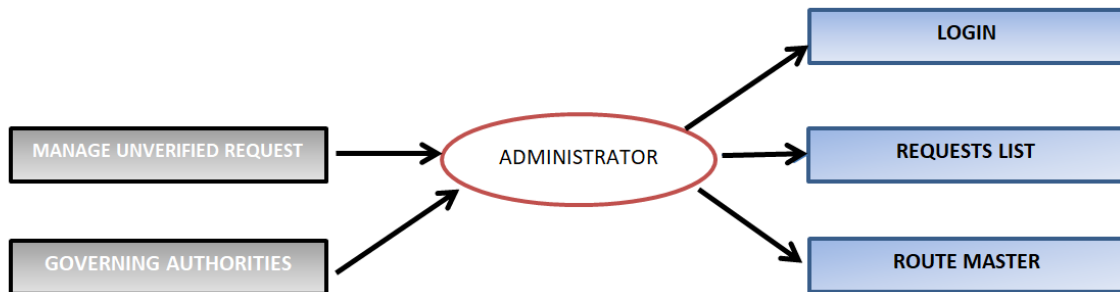


Fig 8 – Administrator DFD

4.5. Database Design

We have designed our database in MYsql , a database management system.

Our database contains 6 tables which are listed below-

- 1)Costumers
- 2)Grequests
- 3)migrations
- 4)Officials
- 5)password_resets
- 6)users

Every table contains relevant information which can be presented in the below given diagram.

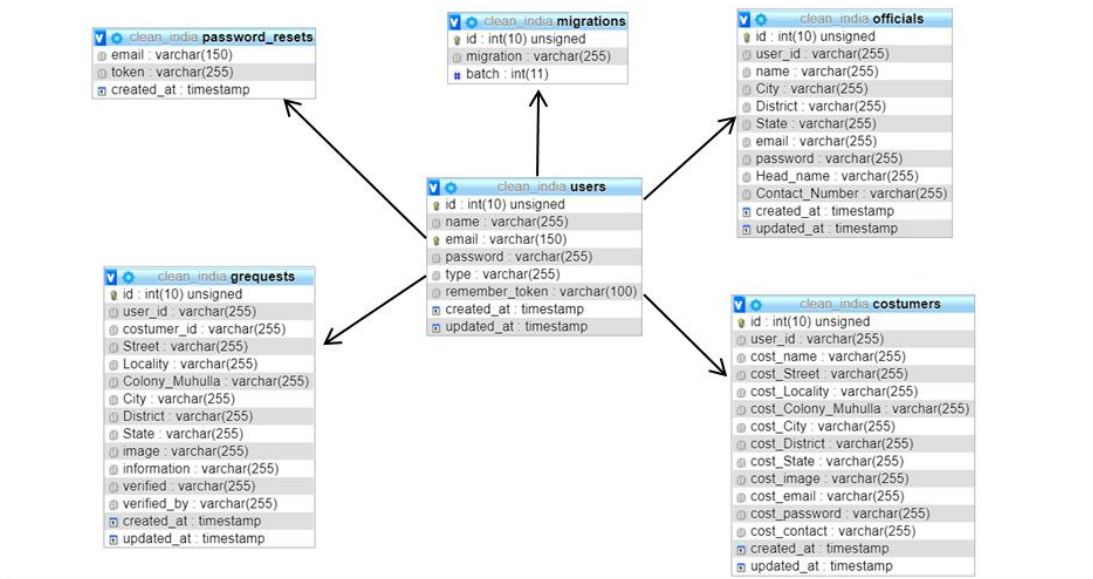


FIG- CLASS DIAGRAM

Fig 9 Class Diagram

4.5.1 Database Details

Every table is associated with each other through many relationships.

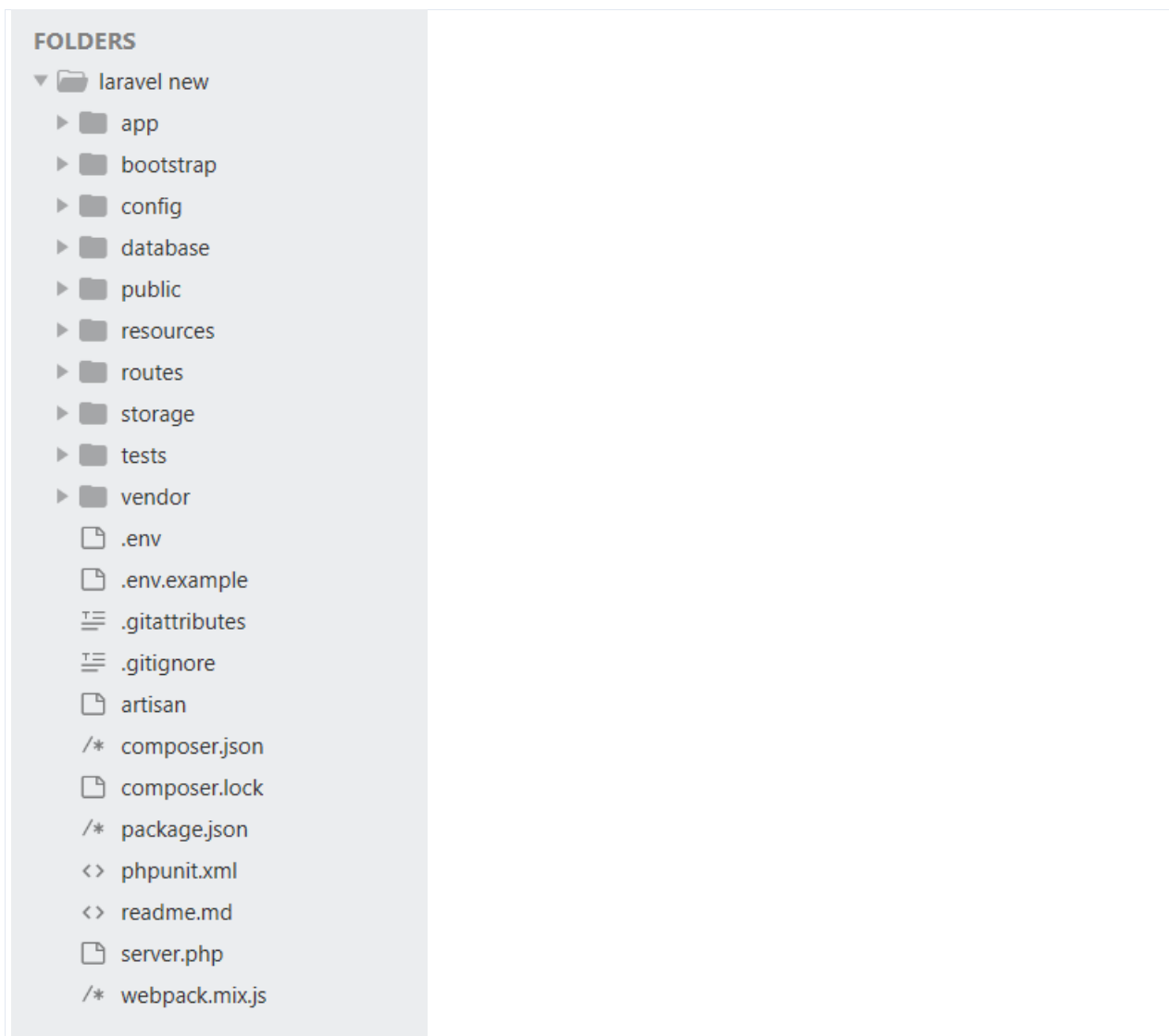
- The table users contain the information of the user and aims towards the login system of the website. Data stored at the time of the registration and on every login data is fetched from table users.
- The table named 'costumers' contains complete details of the users.
A part of this is stored in users table which is required for login purpose. The complete data of every users can be fetched from this table on requirement.
- The table "migrations" contains the record of the migration done by the laravel into the database.
- The table "password_resets" is a table available by the laravel to handle the password forget cases. Data stored in this can help the user in forget password cases.
- The table "grequests" contains all the data of the request registered by the users.
- The table "officials" contains all the data of the government authorities or officials. This data is stored at the time of the registration of official. Every time the complete data of official is required fetched from the same.

5. IMPLEMENTATION

5.1 MVC implementation in Laravel

Before diving into how Laravel implements MVC let us take a look at how requests are handled in Laravel.

When you create a new Laravel project, (you can create one by running the command `laravel new project-name`), the project has the following structure:



There is a file in the `routes/` directory called `web.php`. That file is where you handle the requests when users visit your app. The file looks like this:

```
<?php
Route::get('/', function () {
    return view('welcome');
});
```

In this file, you can route URLs to controllers in your application, for example, what happens when a user goes to 'yourapp.com/home' or 'yourapp.com'.

How MVC is implemented in Laravel applications

Let's take a look at how Laravel uses MVC during development. To do this, let's create a sample Laravel project that displays products for a shop.

To create a new project run the command below in your terminal:

```
$ laravel new clean india project
```

After creating a project, below command is run to terminal to run the laravel server.

```
$ php artisan serve
```

All artisan commands for a Laravel project have to be run inside the root of the Laravel project so you have to `cd` there first before running the command.

This will run your app and you can visit it on `127.0.0.1:8000/`. You should see a welcome view in your browser. Great. To stop the server press `ctrl + c` on your keyboard for Windows, Mac, and Linux.

Models: creating our costumer model

Let us create our first model, M in MVC, for our application. As we have noted before, the model usually, interfaces with a data storage like an SQL database. In Laravel, the Model is usually a class with properties that match the columns in the database.

To create a model in Laravel, run the command in your terminal:

```
$ php artisan make:model costumer
```

When you run this command, Laravel will create a `costumer.php` file in the `app` directory. This will be a PHP class with the name `Product` and it will be the model for our `products` table in the database.

In the `costumer` model, add the `$fillable` property as shown below:

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class costumer extends Model {

    protected $fillable = [
        'cost_name',
        'cost_email',
        'cost_image',
        'cost_password',
    ];

}
```

The `fillable` property is used to represent mass assignable attributes for our model

That is all about models in Laravel, however, as a bonus let's talk about migrations in Laravel.

Migration lets developers make and undo changes to a project's database. Migrations can be used to make managing databases easy and predictable. To create a migration, run the following in your terminal:

```
$ php artisan make:migration create_costumer_table
```

When the command is executed, we should see a new `*_create_costumer_table.php` file in the `database/migrations` directory and we can edit it to have our `products` table schema like this:

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateCostumersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('costumers', function (Blueprint $table) {
            $table->increments('id');
            $table->string('user_id');
            $table->string('cost_name');
            $table->string('cost_Street');
            $table->string('cost_Locality');
            $table->string('cost_Colony_Muhulla');
            $table->string('cost_City');
            $table->string('cost_District');
            $table->string('cost_State');
            $table->string('cost_image');
            $table->string('cost_email');
            $table->string('cost_password');
            $table->string('cost_contact');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('costumers');
    }
}
```

That's all for the migration file. However, before we run the migration we need Laravel to connect to a database. In this tutorial, we will be using SQLite.

As part of the prerequisites mentioned earlier, you need [SQLite installed on your machine](#). To make it possible for Laravel connect to an SQLite database, create a new empty `database/database.sqlite` file.

Next copy the `.env.example` file in the root of your project to `.env` and then in the copied file, replace the following lines:

```
DB_CONNECTION=mysql
DB_DATABASE=clean_india
DB_USERNAME=root
DB_PASSWORD=
```

That is all for our database setup. Now to run the migrations, run the command below in your terminal:

```
$ php artisan migrate
```

Before running the command, you need to have set up your database and set the connection details in your `.env` file in the root of the project.

You can read about how to set up your database [here](#) and you can read more about migrations [here](#).

Controllers: creating our controller

Earlier we mentioned that controllers are responsible for completing user actions and the managing the business logic of our applications. For our make-believe project, we are going to use [Resource Controllers](#).

Laravel resource routing assigns the typical “[CRUD](#)” routes to a controller with a single line of code. For example, you may wish to create a controller that handles all HTTP requests for “photos” stored by your application. – [Laravel documentation](#)

To create a resource controller in Laravel, run the following command:

```
$ php artisan make:controller CostumerController -r
```

The `-r` flag makes it a resource controller and thus creates all the methods required for CRUD operation.

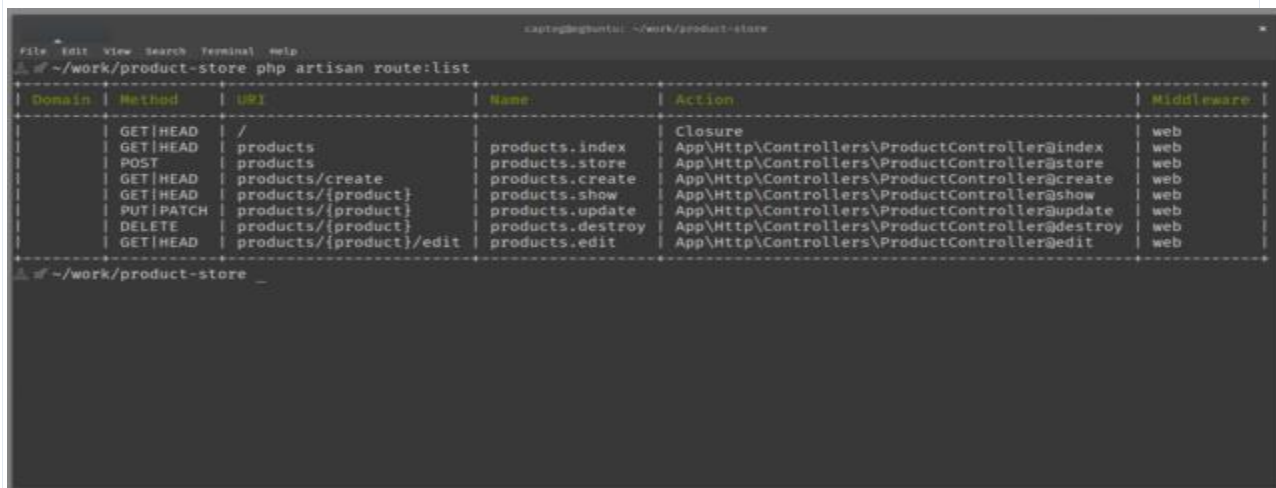
When the command is run, Laravel will create a new file in the `app/Http/Controllers` directory called `CostumerController.php`.

Before we start adding logic to the controller, go to the `routes/web.php` file and add the following route:

```
Route::resource('/costumer', CostumerController);
```

This tells Laravel to create all the routes necessary for a resource controller and map them to the `CostumerController` class.

You can see the list of routes by running the command `php artisan route:list` in your terminal. You get the following result showing the routes and the request types to use when accessing the routes:



Domain	Method	URI	Name	Action	Middleware
	GET HEAD	/		Closure	web
	GET HEAD	products	products.index	App\Http\Controllers\ProductController@index	web
	POST	products	products.store	App\Http\Controllers\ProductController@store	web
	GET HEAD	products/create	products.create	App\Http\Controllers\ProductController@create	web
	GET HEAD	products/{product}	products.show	App\Http\Controllers\ProductController@show	web
	PUT PATCH	products/{product}	products.update	App\Http\Controllers\ProductController@update	web
	DELETE	products/{product}	products.destroy	App\Http\Controllers\ProductController@destroy	web
	GET HEAD	products/{product}/edit	products.edit	App\Http\Controllers\ProductController@edit	web

Looking at the image above, you can see the action that each `URI` is mapped to. This means when a user goes to `127.0.0.1:8000/costumer/create`, the `create` function in the `CostumerController` will process the user's request.

Now let's go to the controller file and update the methods in them with the following logic:

The `create` method:

```
public function create()
{
```

```

        return view('addcostumer');
    }
}

```

The above is for the create (C in CRUD). The controller loads a view (V in MVC) called `create product` and serves that as the response for anytime someone visits the route `/products/create` with a `GET` HTTP method.

The store method:

```

    public function csave(Request $Request){
        $file = $Request->file('image');
        $filename = 'image' . time() . '.' . $Request->image->extension();
        $file->move("upload/", $filename);

        $obj = new user;
        $obj->name = $Request->cost_name;
        $obj->email = $Request->cost_email;
        $obj->password = bcrypt($Request->cost_password);
        $obj->type=2;
        $obj->save();

        $data = new costumer;
        $data->user_id = $obj->id;
        $data->cost_name = $Request->cost_name;
        $data->cost_Street = $Request->cost_Street;
        $data->cost_Locality = $Request->cost_Locality;
        $data->cost_Colony_Muhulla = $Request->cost_Colony_Muhulla ;
        $data->cost_City = $Request->cost_City;
        $data->cost_District = $Request->cost_District;
        $data->cost_State = $Request->cost_State;
        $data->cost_email = $Request->cost_email;
        $data->cost_password = $Request->cost_password;
        $data->cost_contact = $Request->cost_contact;
        $data->cost_image = $filename;
        $created = $data->save();

        if($created){
            return redirect('/home')->with('message', 'User Added Successfully');
        }
    }
}

```

The `csave` method is called when a user sends a `POST` HTTP request to the `/products` endpoint. This logic above gets the data from the request and stores it in the database using the `costumer` model.

To keep the article short, we will limit the controller logic to these three methods. Let's create the views that are loaded by the controller methods above.

Views: creating the projects views

In Laravel, all the views are stored in the `resources/views` directory. Your views usually store the HTML of your page and are the presentation layer of the MVC architecture.

Laravel uses Blade as its templating engine. Blade is pretty much HTML but with some injectable PHP-like syntax. You can read more about [blade here](#).

If you go back to the `routes/web.php` you see it stated that the `welcome` view should be rendered to the user when the `/` is visited. If you visit the webpage URL `http://127.0.0.1:8000/` you will see this page:

Next, let's make the 'Create Product' view. Create a `addcostumer.blade.php` file in the `resources/views` directory of our project. In there add the following:

```
<div class="card"> <div class="card-header"> <h3>
    <strong>Add User</strong></h3> </div> <div class="card-body card-block">

<br>

<form action="{{url('/csave')}}> <div class="form-horizontal">
    <div class="row form-group">
        <div class="col col-md-3"><label for="text-input" class="form-control-label">User
        Name</label></div>
        <div class="col-12 col-md-9"> <input type="text" id="text-input" name="cost_name"
        placeholder="Enter User Name" class="form-control"></div> </div>

        <div class="row form-group">
            <div class="col col-md-3"><label for="text-input" class="form-control-label">Street</label></div>
```



```

<div class="col-12 col-md-9"> <input type="text" id="text-input" name="cost_Street"
placeholder="Enter street Name" class="form-control"></div> </div>

<div class="row form-group"> <div class="col col-md-3"><label for="text-input"
class=" form-control- label">Locality</label></div> <div
class="col-12 col-md-9"><input type="text" id="text-input"
name="cost_Locality" placeholder="Enter Locality" class="form-control">
</div> </div>

<div class="row form-group"> <div class="col col-md-3"><label for="text-input"
class=" form-control- label">Colony/Muhulla</label></div> <div
class="col-12 col-md-9"><input type="text" id="text-input"
name="cost_Colony_Muhulla" placeholder="Enter Colony/Muhulla" class="form-control">
</div> </div>

<div class="row form-group"> <div class="col col-md-3"><label for="text-input"
class=" form-control- label">City</label></div> <div
class="col-12 col-md-9"><input type="text" id="text-input"
name="cost_City" placeholder="Enter City" class="form-control">
</div> </div>

<div class="row form-group"> <div class="col col-md-3"><label for="text-input"
class=" form-control- label">District</label></div> <div
class="col-12 col-md-9"><input type="text" id="text-input"
name="cost_District" placeholder="Enter District" class="form-control">
</div> </div>

<div class="row form-group"> <div class="col col-md-3"><label for="text-input"
class=" form-control- label">State</label></div> <div
class="col-12 col-md-9"><input type="text" id="text-input"
name="cost_State" placeholder="Enter State" class="form-control">
</div> </div>

<div class="row form-group"> <div class="col col-md-3"><label for="text-input"
class=" form-control- label">Email</label></div> <div
class="col-12 col-md-9"><input type="email" id="text-input"
name="cost_email" placeholder="Enter Email" class="form-control">
</div> </div>

<div class="row form-group"> <div class="col col-md-3"><label for="text-input"
class=" form-control- label">Enter Password</label></div> <div
class="col-12 col-md-9"><input type="password" id="text-input"
name="cost_password" placeholder="Enter password " class="form-control">
</div> </div>

<div class="row form-group">
<div class="col col-md-3"><label for="text-input" class=" form-control- label">Enter
Contact person Number </label></div>
<div class="col-12 col-md-9"><input type="number" id="text-input"
name="cost_contact" placeholder="Enter Contact Person Number " class="form-control">
</div> </div>

```

```

<div class="row form-group">
  <div class="col col-md-3">
    <label class="form-control-label">Picture of User</label>
  </div>
  <div class="col-12 col-md-9">
    <input type="file" name="image" class="form-control-file">
  </div>
</div>

<input type="submit" value="Submit"> </form> </div>

</div>

```

This view above is a simple form that collects and submits requests to create products. When the form is submitted, a **POST** request is made to the `/costumers` route of the application which is handled by the `store` method in our `CostumerController`.

Here is how the `/costumers/addcostumer` route will look after adding the view and visiting the route:

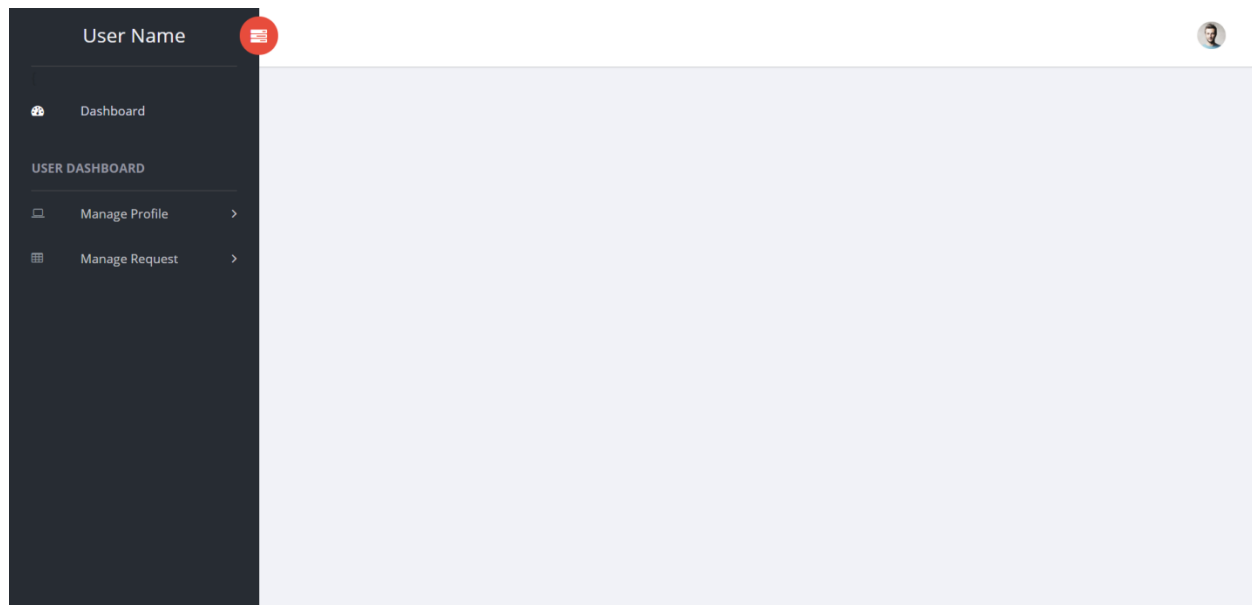
Add User

User Name	<input type="text" value="Enter User Name"/>
Street	<input type="text" value="Enter street Name"/>
Locality	<input type="text" value="Enter Locality"/>
Colony/Muhulla	<input type="text" value="Enter Colony/Muhulla"/>
City	<input type="text" value="Enter City"/>
District	<input type="text" value="Enter District"/>
State	<input type="text" value="Enter State"/>
Email	<input type="text" value="Enter Email"/>
Enter Password	<input type="text" value="Enter password"/>
Enter Contact person Number	<input type="text" value="Enter Contact Person Number"/>

5.2 User Functionality

Here, we are going to list the features that a user can be facilitated.

User can manage all its features through its dashboard which looks like as-



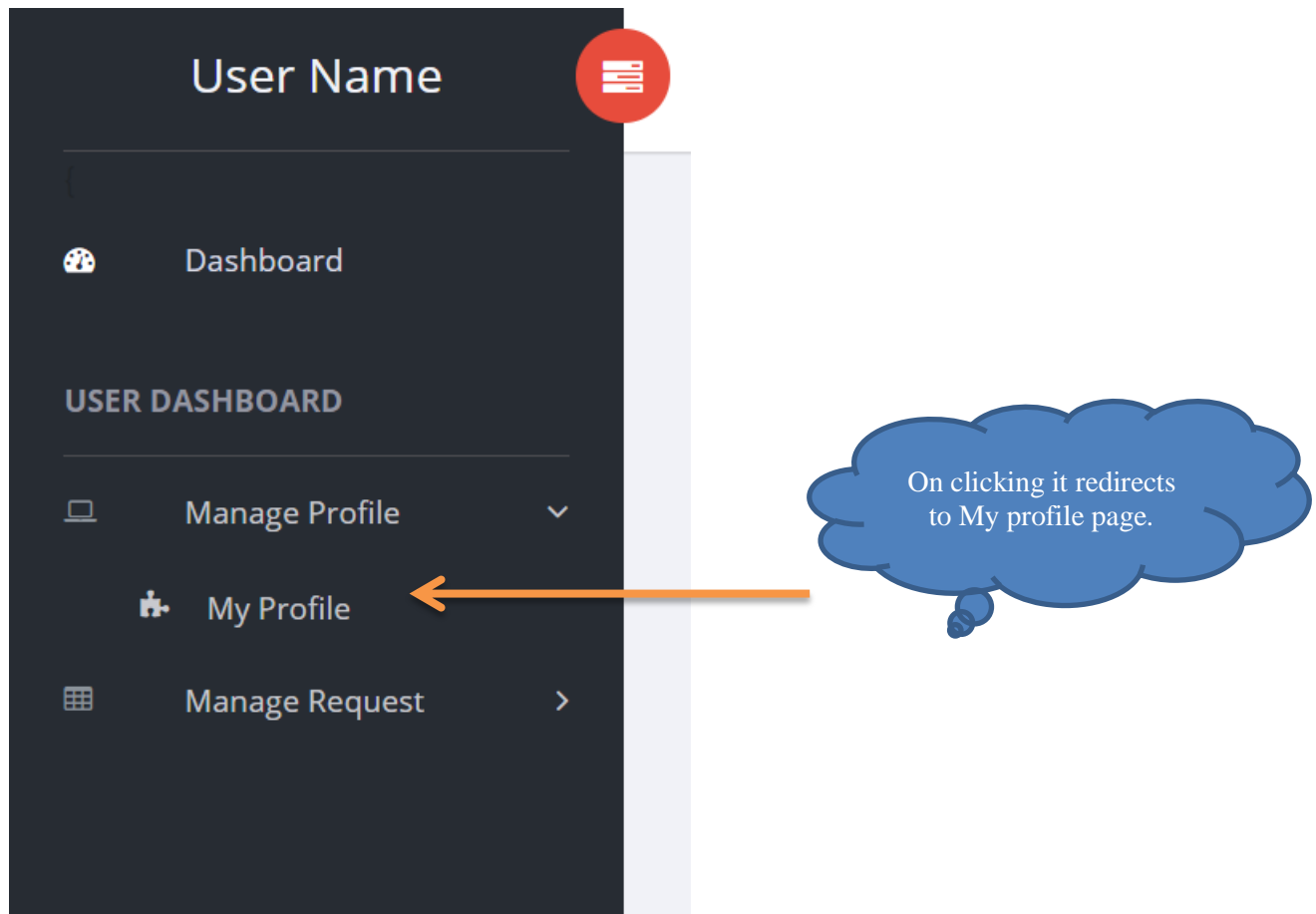
1. Login/Signup of user can be done through login/signup form appear on clicking of login link.

A screenshot of a web application's login page. The page has a light blue background. At the top left, the word 'Laravel' is displayed. At the top right, there are links for 'Login' and 'Register'. In the center, there is a white login form box with a title 'Login'. Inside the form, there are two input fields: 'E-Mail Address' and 'Password'. Below the password field is a checkbox labeled 'Remember Me'. At the bottom of the form, there is a blue 'Login' button and a link that says 'Forgot Your Password?'. The entire form is set against a light blue background.

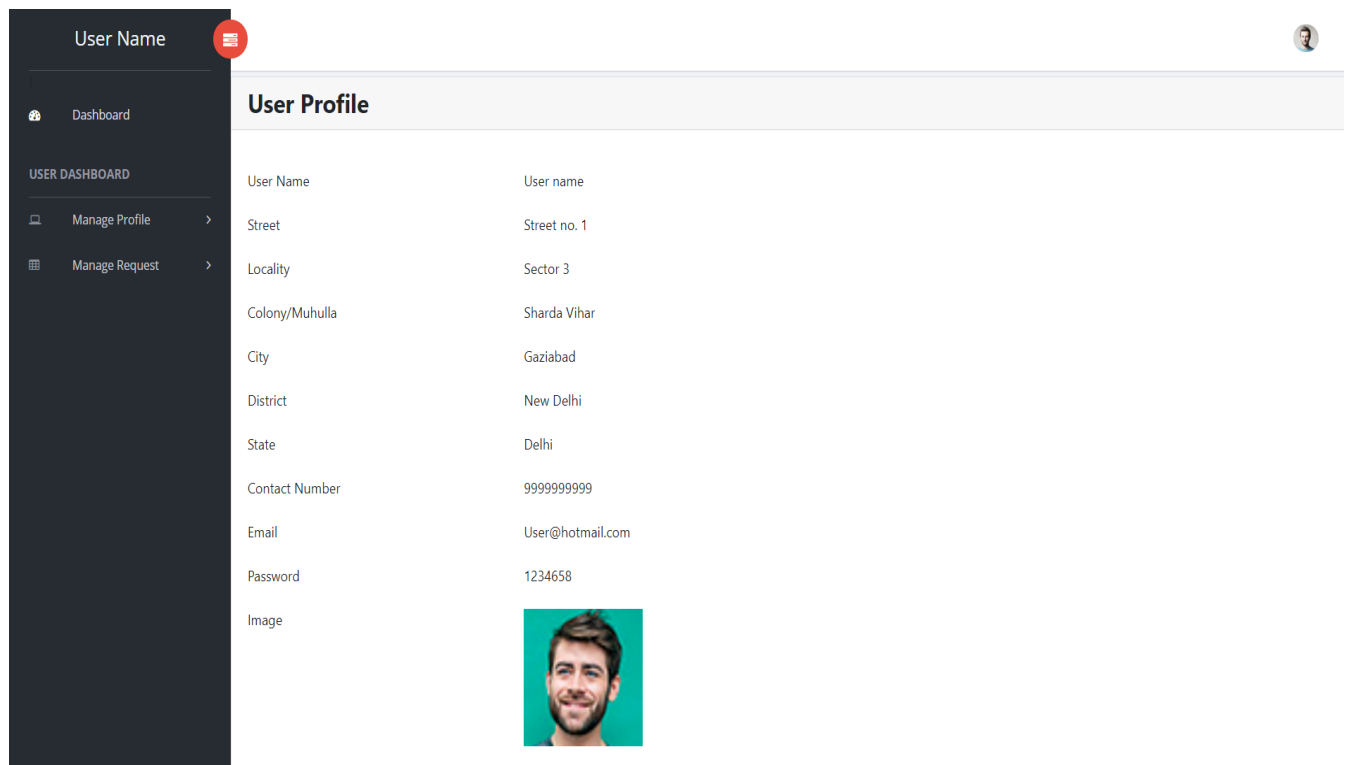
In this login page, user have to put the registered email Id and password which inserted at the time of sign up form. The combination of email and password worked as login credentials here.

2. Profile of user

User can also check his/her profile by clicking on the my profile section in the sidebar.

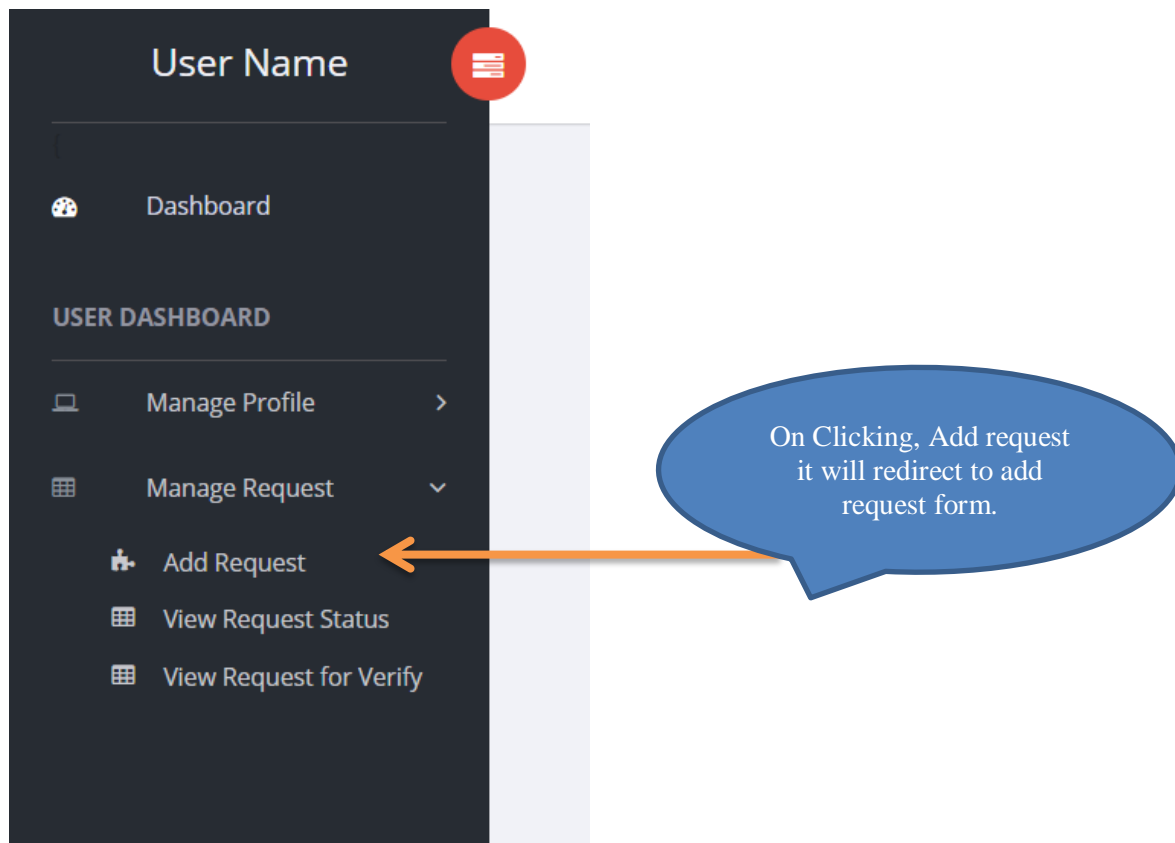


My clicking on the My profile section, we redirects to my profile page where user's profile is shown as given below-



2.Add a request for garbage collection.

Most important feature of our project is to register the request for garbage collection. For accessing this feature user just have to click on the manage request button on sidebar, on clicking it will redirect to the add request form.



After clicking on the add request button on sidebar, it will redirect to the add request form.

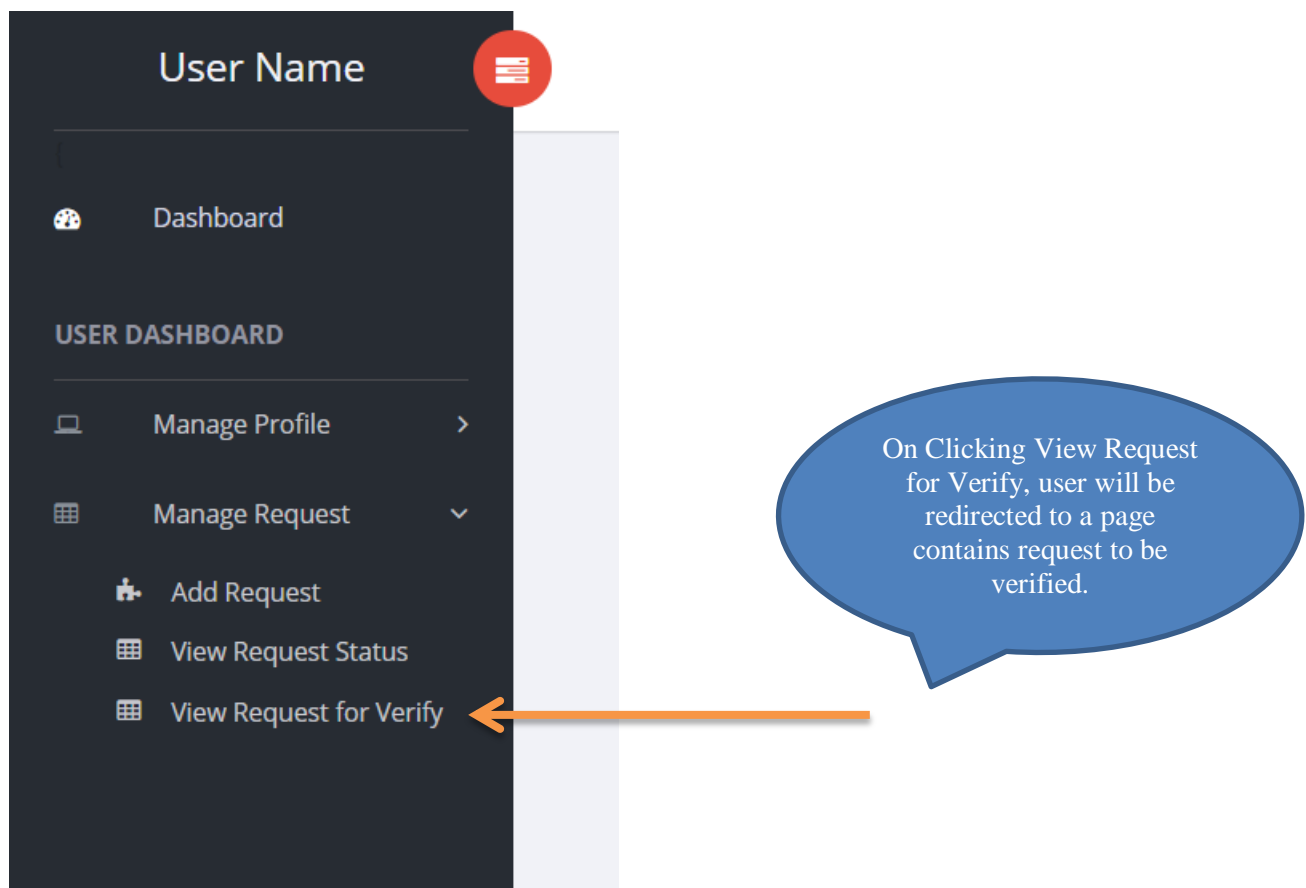
The image shows a form titled 'Add Request' with a light gray header. The form contains several input fields and a submit button. The fields are labeled on the left and have placeholder text in the input boxes on the right. The labels and their corresponding input fields are: 'Street' (placeholder: 'Enter street Name'), 'Locality' (placeholder: 'Enter Locality'), 'Colony/Muhulla' (placeholder: 'Enter Colony/Muhulla'), 'City' (placeholder: 'Enter City'), 'District' (placeholder: 'Enter District'), 'State' (placeholder: 'Enter State'), 'Image' (placeholder: 'Choose File' and 'No file chosen'), and 'Information' (placeholder: 'Enter important Information'). At the bottom left of the form is a 'Submit' button.

By filling this form, your request will be registered and you will be redirected to the homepage.

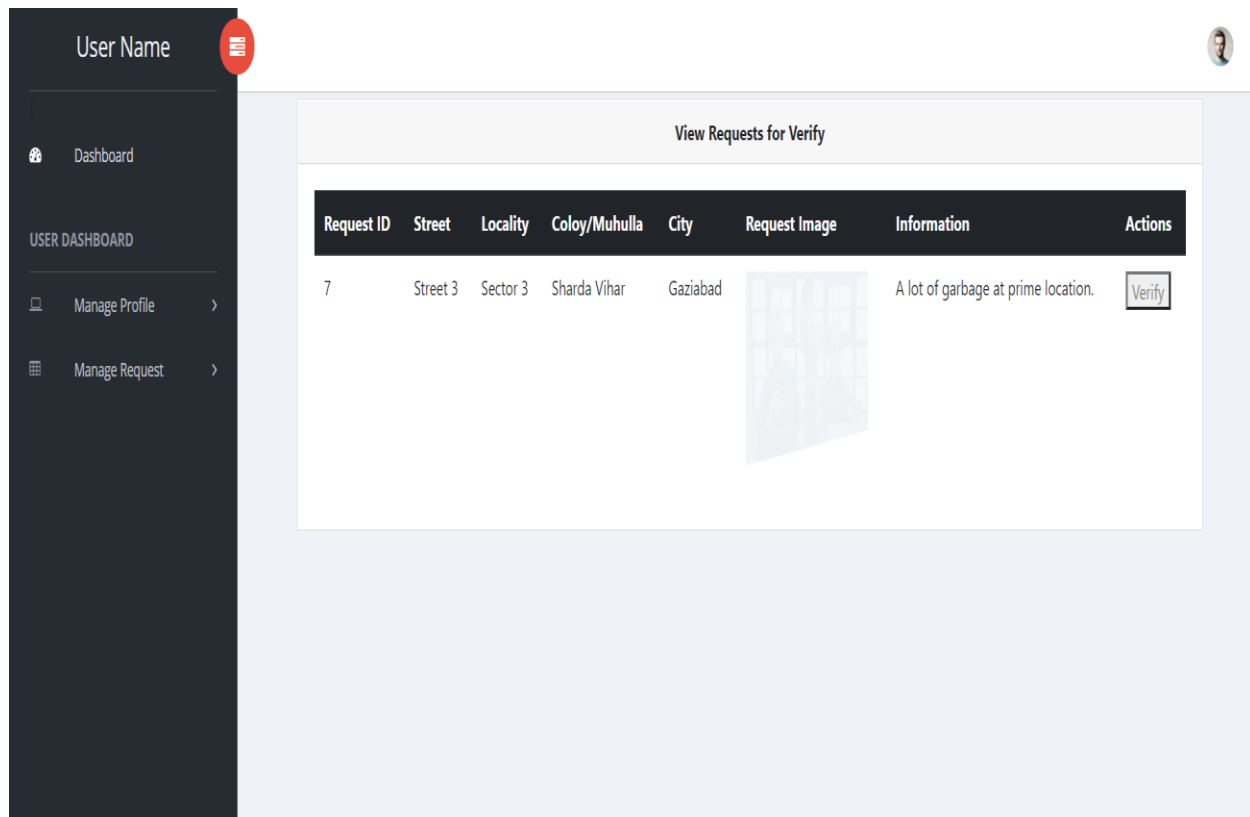
3.Verifying the Request.

First of all, it is essential to clarify the concept behind the verification of the request. Assume USER 1 has registered a request for garbage gathering so this request will be verified by another user of that locality.

For verifying the request, there is a button inside manage requests section in user sidebar.



By clicking on this link, user will be redirected to the page contain requests to be verified, which are registered by the people living near to the USER 1.



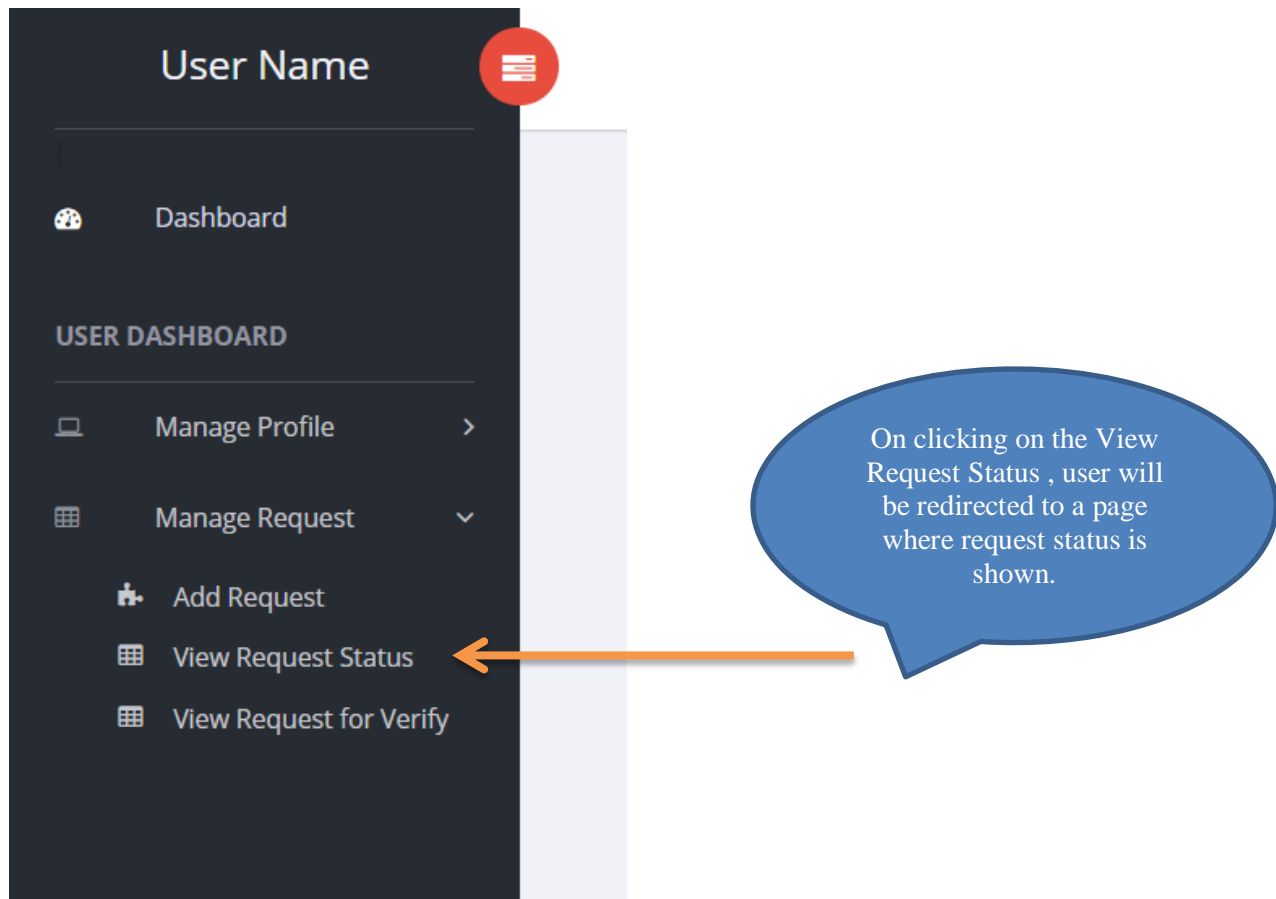
All the required details are given on the page. By following the information we can verify the request.

If the request is genuine then user can click on the verify button hence requested get verified.


4. Check Request Status

There is also a facility of checking the request status whether it is verified or not.

This feature is accessed through sidebar option which is given in below listed image-



After clicking on the View Request Status, user will be redirected to a page where the requests registered by the user are mentioned in proper sequencing.
The redirected page will be like as shown in image below-

View Requests for Verify								
Request ID	Street	Locality	Coloy/Muhulla	City	Request Image	Information	Verified	Verified by
7	Street 3	Sector 3	Sharda Vihar	Gaziabad		A lot of garbage at prime location.	Not Verified	

The verified column present here will let the user know about the current status of the request in which it holds its presence.

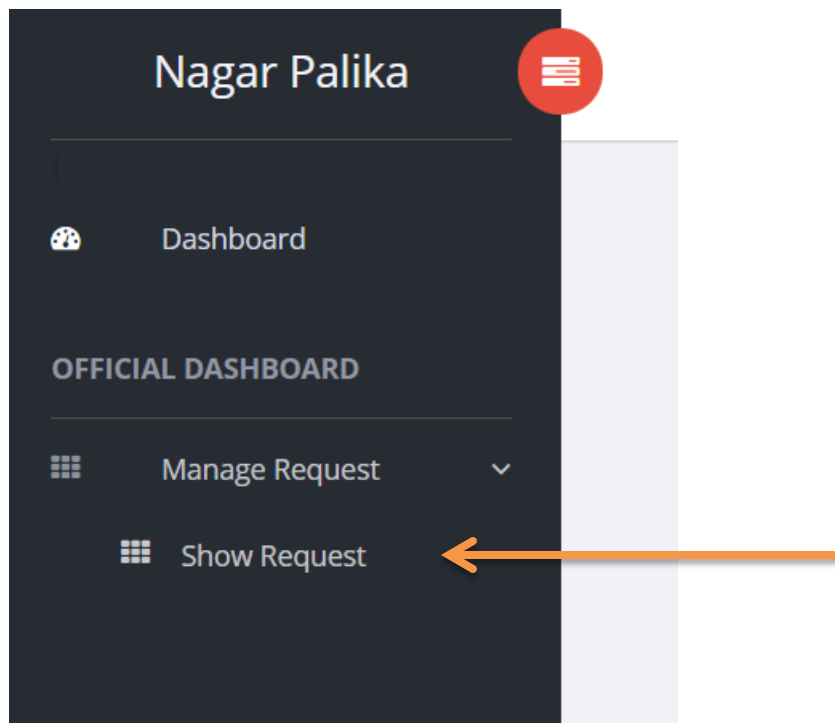
If the request will be verified then the verified column will contain tag “verified” and verified by column will contain the name of the people who verified it.

5.3 Government Authorities/Officials Functionality

As we have discussed the user portal of our project, after the registration of the request, it will be verified by other user and finally it reached to the respective authority of the city or district to take the relevant action for the request.

Government authorities cannot register themselves on our website they will be registered through our admin of the website due to security concerns this step is taken.

Below is government authorities dashboard –

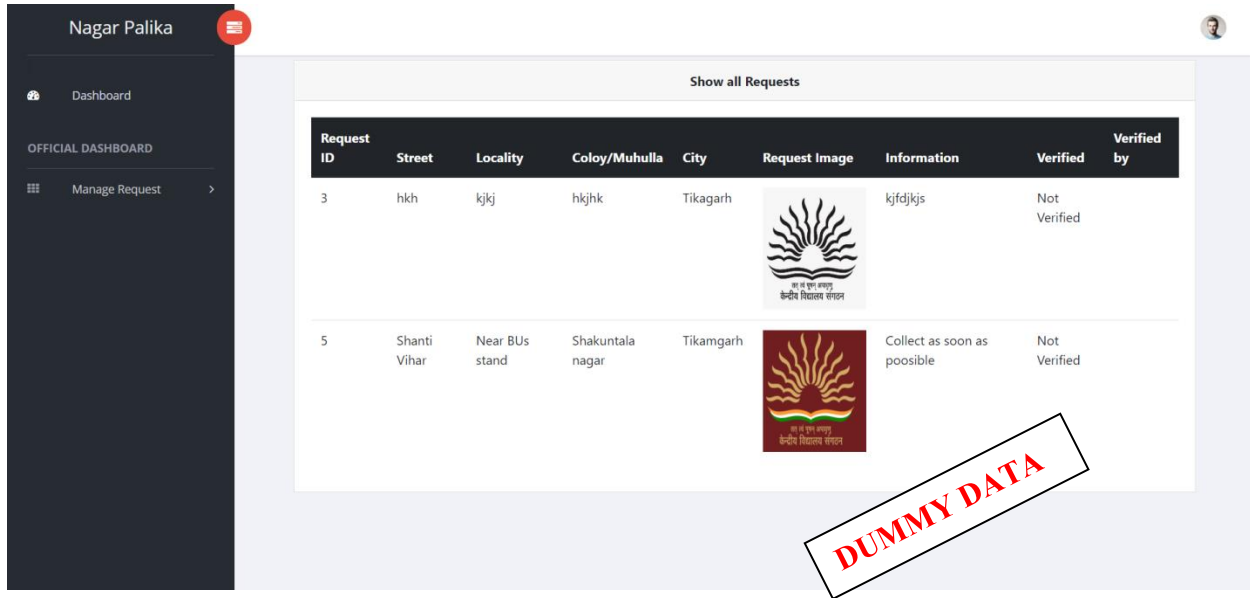




There is a feature to show requests of their regions which are verified by the user except of the registering user.

By clicking on the link “Show request ” official will be redirected to the page containing a list of requests of their region with all corresponding details required by the authority for taking the requisite action.

If any further detail is needed then the authority can contact the admin for details.

Hence, from our view all the necessary details are provided in the below shown section as Show request page by officials.



Request ID	Street	Locality	Coloy/Muhulla	City	Request Image	Information	Verified	Verified by
3	hkh	kjkj	hkjhk	Tikagarh		kjfdjkjs	Not Verified	
5	Shanti Vihar	Near BUs stand	Shakuntala nagar	Tikamgarh		Collect as soon as poossible	Not Verified	

DUMMY DATA

In this page, all requests are shown in proper order with proper details and along with a column that the request is verified or not.

The governing authority can take action on verified requests as well as on non verified requests according to their norms.

Once the details of the officials are filled then it cannot be modified by themselves, only can be modified through Administrator of the project.

5.4 Administrator Functionality

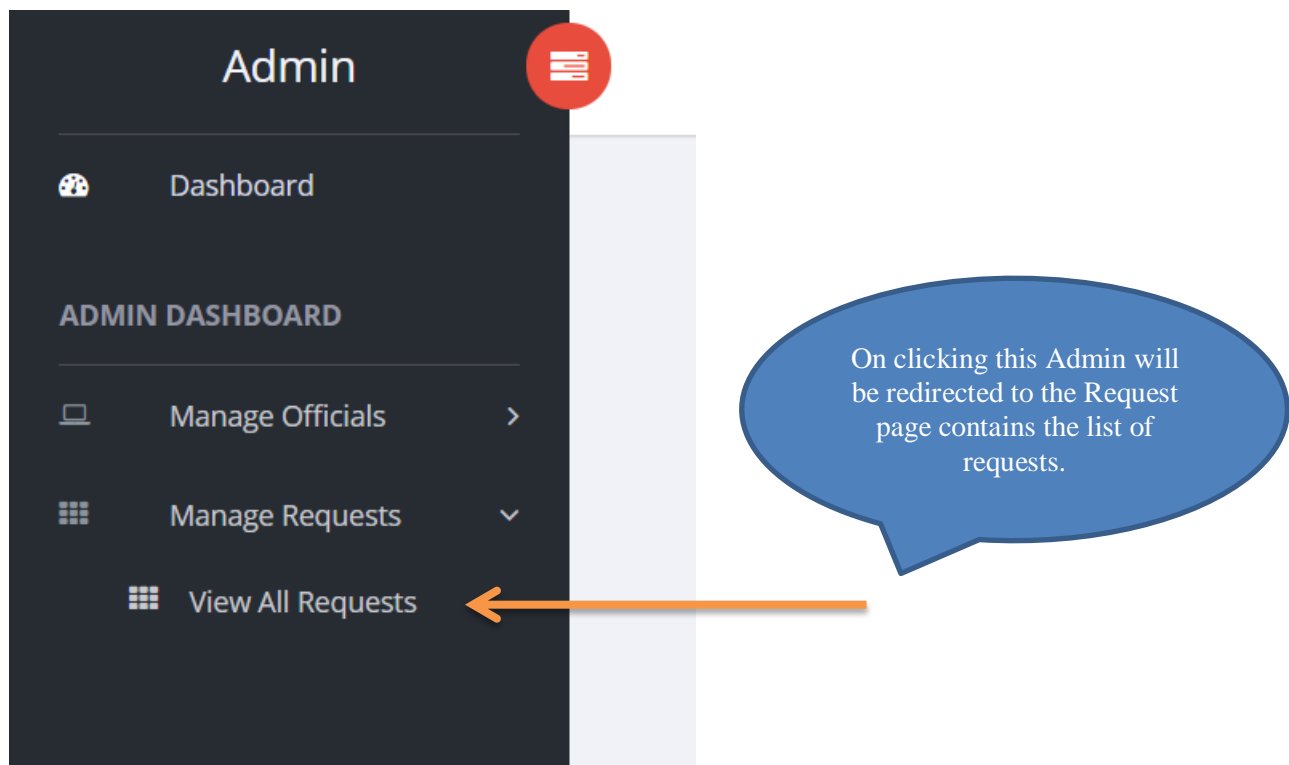
Our website is managed by the the special user called Administrator. For managing our website contents we have designed Admin panel. Admin have all the grants to access the data.

Administrator functionalities are given below-

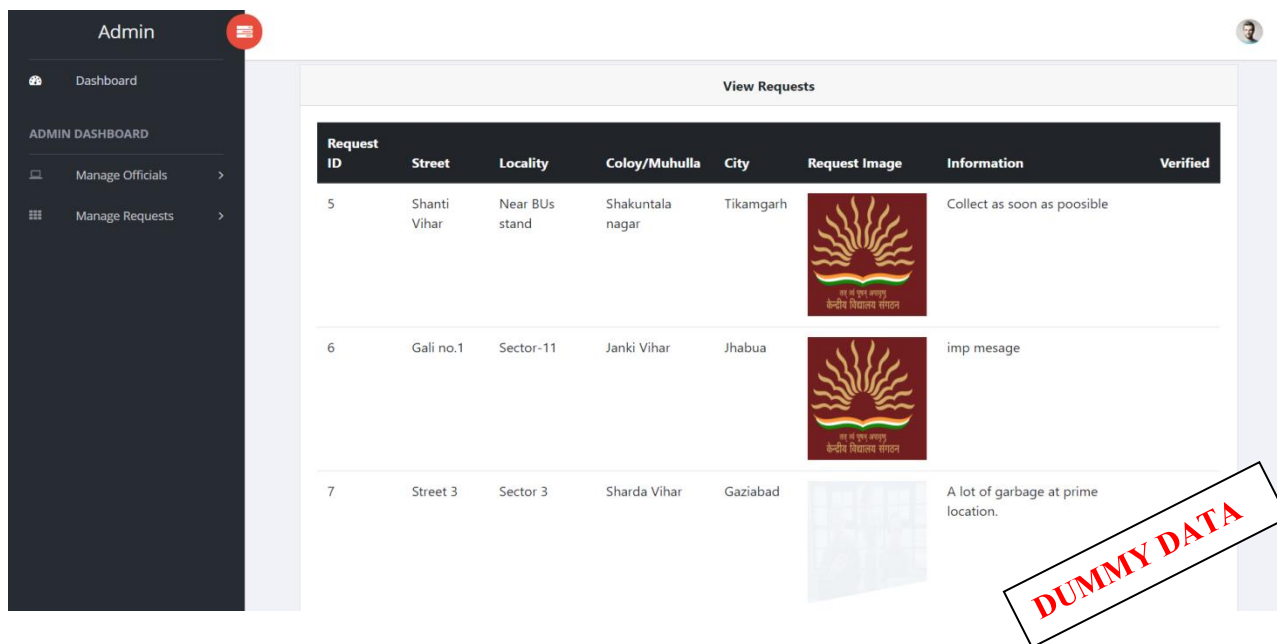
1.List of Requests




Admin can see all the requests registered by users, irrespective of the locality and verification status.

List contains all the requests registered by the users. For this admin have to click on the link given in the sidebar of the admin panel.



By clicking in this link, admin is redirected on the request page which contain all the requests which are registered by user.



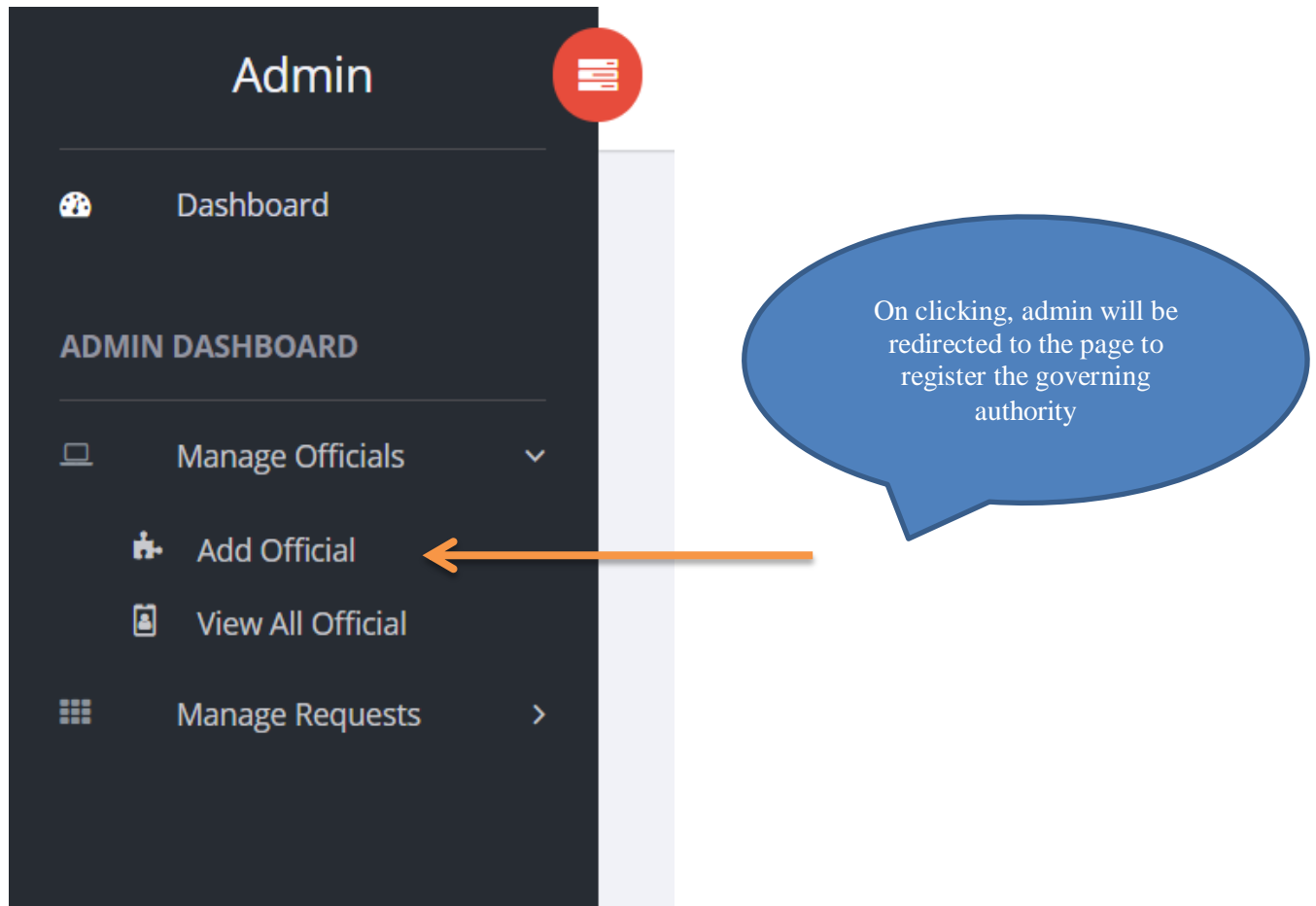
Request ID	Street	Locality	Coloy/Muhulla	City	Request Image	Information	Verified
5	Shanti Vihar	Near BUs stand	Shakuntala nagar	Tikamgarh		Collect as soon as possible	
6	Gali no.1	Sector-11	Janki Vihar	Jhabua		imp message	
7	Street 3	Sector 3	Sharda Vihar	Gaziabad		A lot of garbage at prime location.	

Above given is DUMMY DATA, but the prime motive of the image is to show the format of the requests that will be shown to admin.

2. Register Government Authorities/Officials

Nagar Palika/Nagar nigam or government authorities are registered through administrator.

For registration, administrator have to click on the link given in the sidebar of the admin portal.



On clicking this given link in the sidebar, admin will be redirected to a form for registering the government authority.

Form contain all the necessary details which you can see in below listed image.

The registered email and password set at the time of the registration will be used for the login as login credentials.

Combination of registered email and password will be only able to login in the website, in case of any, mishappening these can be updated by the administrator.

The screenshot shows an Admin dashboard with a sidebar on the left and a main content area. The sidebar has a header 'Admin' with a red notification icon, a 'Dashboard' link, and an 'ADMIN DASHBOARD' section with 'Manage Officials' and 'Manage Requests' links. The main content area is titled 'Add Official' and contains a form with the following fields:

- Official Name:
- City:
- District:
- State:
- Email:
- Enter Password:
- Enter Head Person Name:
- Enter Official Contact Number:

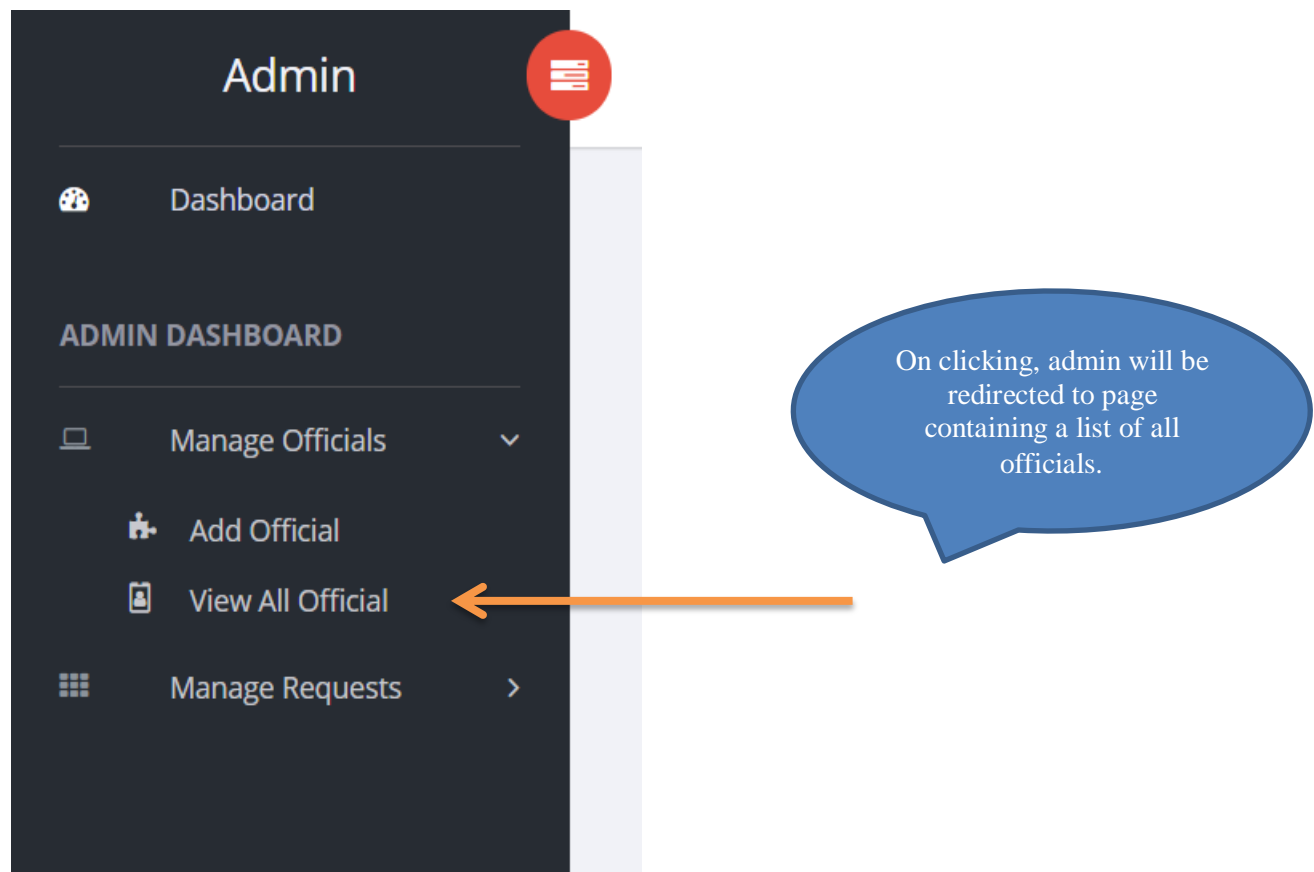
A 'Submit' button is located at the bottom left of the form.

This is the form contains details to be filled at the time of registration of government authority. The fields given in form are all required details, if any of them remains vacant then the form will not be submitted.

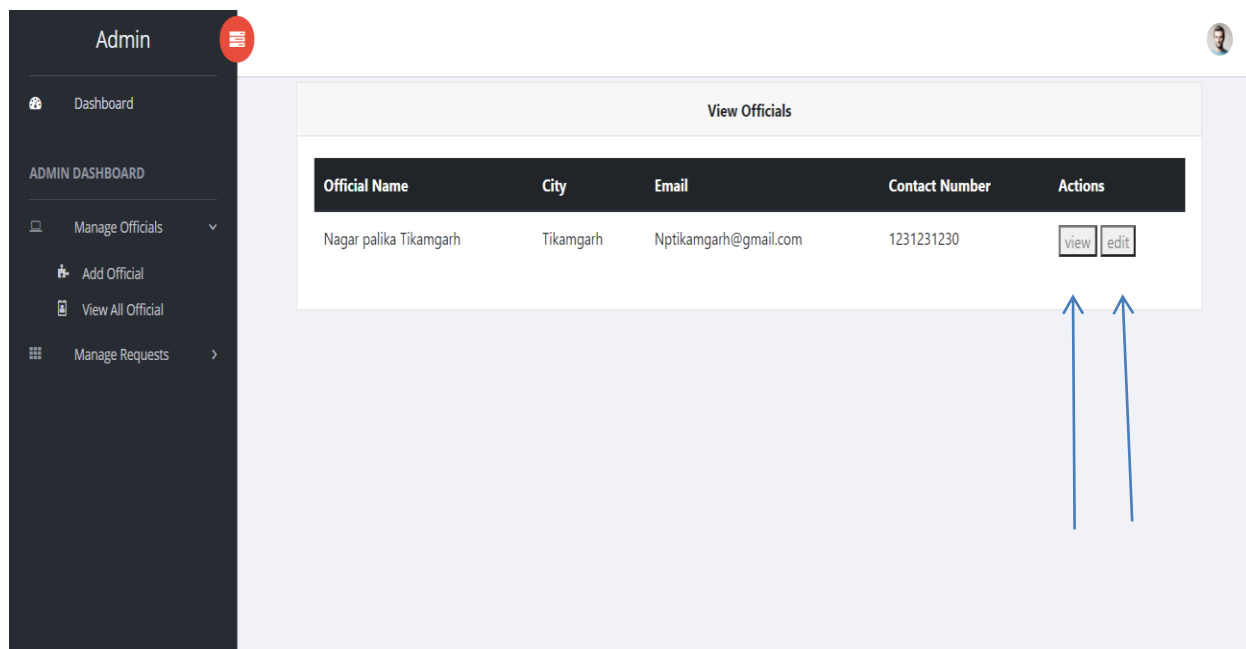
3View all government authorities/Officials registered

There is a facility to see all the registered government authorities at one page along with their all details.

For accessing this page, admin has to click on the VIEW ALL OFFICIAL link given in the sidebar.



After clicking on this link, admin will be redirected to the page containing the list of the officials with some action buttons which are used to manipulate their content.



Above is given a list of officials which are registered, you can also see two action buttons as View and edit.

i)VIEW –This button is used to see the complete details of the official.

By clicking on the view button a new page is opened as containing complete details of the official.

Admin

Dashboard

ADMIN DASHBOARD

Manage Officials >

Manage Requests >

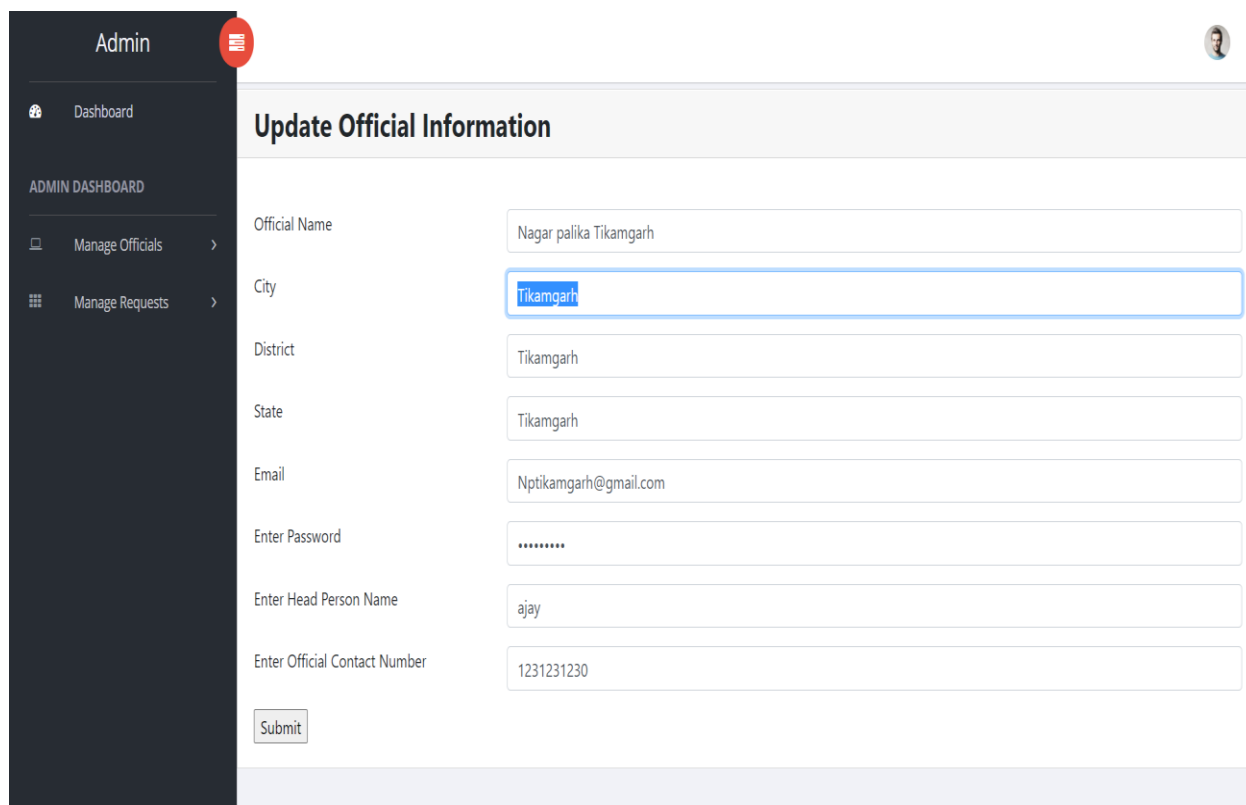
View Official Details

Official Name	Nagar palika Tikamgarh
City	Tikamgarh
District	Tikamgarh
State	Tikamgarh
Email	Nptikamgarh@gmail.com
Password	123123123
Head Person Name	ajay
Contact Number	1231231230

ii)EDIT – This button provides the facility of updating the existing information of the official or Government authorities.

By clicking this button , a new page is opened contains a form filled by existing values and if any of the values are needed to be updated then it can be done by editing the form.

Below given form shows the format of the page-



The screenshot shows an Admin portal interface. On the left is a dark sidebar with the 'Admin' title and a red notification badge. The sidebar menu includes 'Dashboard', 'ADMIN DASHBOARD', 'Manage Officials', and 'Manage Requests'. The main content area is titled 'Update Official Information' and contains a form with the following fields: 'Official Name' (Nagar palika Tikamgarh), 'City' (Tikamgarh, highlighted with a blue border), 'District' (Tikamgarh), 'State' (Tikamgarh), 'Email' (Nptikamgarh@gmail.com), 'Enter Password' (masked with dots), 'Enter Head Person Name' (ajay), and 'Enter Official Contact Number' (1231231230). A 'Submit' button is located at the bottom left of the form.

Official Name	Nagar palika Tikamgarh
City	Tikamgarh
District	Tikamgarh
State	Tikamgarh
Email	Nptikamgarh@gmail.com
Enter Password	*****
Enter Head Person Name	ajay
Enter Official Contact Number	1231231230

Submit

As you can see city field is going to be edited, similarly all fields are editable so the arbitrary field can be updated by the administrator.

Hence, these are all the features of the admin portal through which we can manage our website.

CONCLUSION

We have developed a solution for collecting garbage from the city on the populace request. Our platform will provide a source to those who register on our website and just raise a request for garbage collection which is gathered in their livelihood.

Our website is connected with the government authorities which are responsible for garbage collection with the city/District. Our user only have to register on our website and register a request for garbage with the picture of garbage and location.

After the registration of request, the people of that area will get notified about that request registered by that user to cross verify. The people of that area can verify the request if it had genuineness otherwise it will be marked as fake by the people.

After the verification of the request, it has been send to the nagar palika/nagar nigam the respective governing authority to take action for that request.

So now registering a request is on fingertips and can solve the problem of garbage in your locality.

Now people don't have to go government offices for garbage collection stuffs just use our platform and get rid off your nearby garbage.

Our website is light in size and having an attractive interface which leads to simply access of the website. There is no need of professionalism to operate our website, hence a common people can use it without any problem.

FUTURE WORK

We have some future plans for our project which will enhance the feasibility, easy accessibility, attractive design and many more.

First of all, we will design an automated system for debar the fake requests and warn the user for doing the same. If the user do it thrice then the account of the user will be blocked after that if user tries with another account then it will be punishable offence and treated by the judiciary of the country.

If someone tries to make a fake profile then it will also be detected in our future versions.

We are also planning for the mobile application so that every user can use it any time without access to the website/browser.

We will also try to gain some authentication api to implement in our website so that identity theft can be taken care of it.

We also planned to implement a location api which will automate the manual task of address filling and user will then register the request in a very small amount of time.

Currently, we had not launched our website, but in very few time we will launch it so that people can use it and facilitate themselves.

Our website having a service based intention so that there is no need of any economic support or heavy funding.

REFERENCE

1. <https://laravel.com/docs/5.4>
2. <https://stackoverflow.com/>
3. <https://medium.com/>
4. <https://www.w3schools.com/>
5. <https://getbootstrap.com/>
6. <https://material-ui.com/>

APPENDICES

Appendices consist of all the code scripts that have been used during this project. Below there is a reference of all of them, explaining what they are intended to do:

- addcostumer.blade.php = This file contains a form for registration of the user.
- addofficial.blade.php = This file contains a form to add official through admin portal.
- addrequest.blade.php = This file contains a form to add a request by the user.
- cost.profile.blade.php = This file contains a form type frontend code for display user Profile.
- editofficial.blade.php = This file having content to update the official details.
- home.blade.php = This file contains the landing page of the home page.
- master.blade.php = This file contains Html codes of sidebar and navbar as a skeleton.
- requestforverify.blade.php = This file having code for list of requests to verify.
- requeststatusbyuser.blade.php = This file contain code for the list request with their present status as verified or not.
- showofficial.blade.php = This file contain the list of the officials shown to admin.
- showrequest.blade.php = This file contain code for complete information if the particular request.
- showrequestadmin.blade.php = This file contain code for a list of requests shown to admin.
- showrequestofficial.blade.php = This file contain list of requests shown to official of their particular area.
- viewofficial.blade.php = This file shows complete information about particular official.
- welcome.blade.php = This file contains code that is shown as default page on opening.
- header.blade.php = This file contains header of the pages and includes same in every file.
- sidebar.blade.php = This file contains sidebar which changes according to the user's access.