

PART 1

TECHNOLOGY FRAMEWORKS

1.1 LARAVEL



1.1.1 LARAVEL - OVERVIEW

Laravel is an open-source PHP framework, is robust as well as simple to learn. In simpler form, it is an open-source web application development framework written in PHP. It follows a model-view-controller design pattern. Taylor Otwell created Laravel and released under MIT license. Laravel reuses the existing elements of various frameworks which aid in building a web application.

Laravel offers a rich collection of functionalities which consolidates the primary characteristics of PHP frameworks like Code Igniter, Yii and different programming languages like Ruby on Rails. Laravel has a vibrant set of features which will heighten the pace of web development. Laravel is the most successful elected web development aloft other PHP based MVC frameworks due to its integrity, execution, scalability, and features.

Laravel is an MVC framework with bundles, migrations, and Artisan CLI. Laravel offers a robust set of tools and an application architecture that incorporates many of the best features of frameworks like CodeIgniter, Yii, ASP.NET MVC, Ruby on Rails, Sinatra, and others.

Laravel is an Open Source framework. It has a very rich set of features which will

boost the speed of Web Development. If you familiar with Core PHP and Advanced PHP, Laravel will make your task easier. It will save a lot time if you are planning to develop a website from scratch. Not only that, the website built in Laravel is also secure. It prevents the various attacks that can take place on websites.

Laravel is a compelling model view controller (MVC) architectural pattern PHP framework, an open-source web application development intended for developers who demand an uncomplicated and rich toolkit to build full-featured web applications. Laravel is a PHP framework which makes it effortless for you to produce professional web application by following refined coding standards and architectural pattern. This concise tutorial illustrates the basics of the Laravel framework.

Why Laravel?

- Assists professional and advanced web development exercises.
- Promotes rapid as well as reliable web application development
- Makes development, deployment, and maintenance flexible and pleasant.
- Inherent Syntax.
- An elegant set of convenient and advanced built-in features.
- Laravel is the Utmost promising PHP based MVC framework.
- Pretty adequately documented and has the large community of active members.
- And the most desirable and promising thing is it is easy to learn and understand.

Advantages of Laravel

- Built-in Libraries.
- Built-in CLI.
- Built-in Template engine.

- Modular.
- Migration System.
- Error plus Exception Handling.
- Test Driven Development (TDD).
- Security.
- Follows MVC Architecture.
- Built-in ORM (object-relational mapping).

Features of Laravel

- Class Auto loading
- IOC container
- Migration
- Query builder
- Artisan console
- Database Seeding
- Unit-Testing
- Application Logic
- Automatic Pagination
- Form Pagination
- Restful Controllers
- Reverse Routing
- The Eloquent ORM
- View Composers
- Form request
- Bundles

Composer

A composer is a tool, incorporates all the dominions and libraries. It enables a user to generate a project concerning the specified framework (for instance, those adopted in Laravel installation). Third party libraries can be installed efficiently with the help of composer.

Artisan

Command Line Interface Utilized in Laravel Is Named Artisan. It Comprises A Collection of Commands Which Aids in Developing A Web Application. These Commands Are Consolidated from The Symphony Framework, Appearing in Add-On Hallmarks in Laravel 5.1 (Latest Version of Laravel).

1.1.2 LARAVEL – INSTALLATION

LARAVEL PREREQUISITES

- PHP >= 5.6.4
- OpenSSL PHP Extension
- Tokenizer PHP Extension
- Mbstring PHP Extension
- XML PHP Extension
- Xampp or Wampp server installed and configured
- PDO PHP Extension

Step 1:

Visit the below-mentioned URL and download composer to install it on your system.

<https://getcomposer.org/download/>

Step 2:

After downloading composer, open the command prompt browse to the web server root directory or localhost folder.

```

Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\wamp\www\laravel>php artisan --version
Laravel Framework version 5.1.23 (LTS)

C:\wamp\www\laravel>cd\

C:\>composer

Composer version 1.0-dev (c7ed232ef42c2bd63cdba057b6c7c8043b37cd5a) 2015-10-29 09:52:59

Usage:
  command [options] [arguments]

Options:
  -h, --help                Display this help message
  -q, --quiet               Do not output any message
  -V, --version             Display this application version
      --ansi                Force ANSI output
      --no-ansi             Disable ANSI output
  -n, --no-interaction      Do not ask any interactive question
      --profile             Display timing and memory usage information
  -d, --working-dir=WORKING-DIR If specified, use the given directory as working directory.
      --verbose             Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug

```

Step 3:

Of course, first you will need a fresh installation of the Laravel framework. You may use the Homestead virtual machine or the local PHP environment of your choice to run the framework. Once your local environment is ready, you may install the Laravel framework using Composer:

composer global require "Laravel/installer"

```

[ahmed92]:public_html$ composer global require "laravel/installer"
Changed current directory to /home/master/.composer
You are running composer with xdebug enabled. This has a major impact on runtime performance. See https://getcomposer.org/xdebug
Using version ^1.3 for laravel/installer
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
- Installing symfony/process (v3.2.0)
  Downloading: 100%
- Installing psr/log (1.0.2)
  Loading from cache

```

This command will install Laravel Installer. The most important benefit of Laravel Installer is the ease of starting new projects. A new Laravel project could be easily created by a single, easy-to-remember command.

command to create a new project: **Laravel new <project name>**

```
C:\xampp\htdocs>laravel new blog
Crafting application...
Loading composer repositories with package information
Installing dependencies (including require-dev) from lock file
- Installing jakub-ondarka/php-console-color (0.1)
  Loading from cache
- Installing vlucas/phpdotenv (v2.4.0)
  Loading from cache
- Installing symfony/polyfill-mbstring (v1.3.0)
  Loading from cache
- Installing symfony/var-dumper (v3.1.7)
  Downloading: 100%
- Installing symfony/translation (v3.1.7)
  Downloading: Connecting...
```

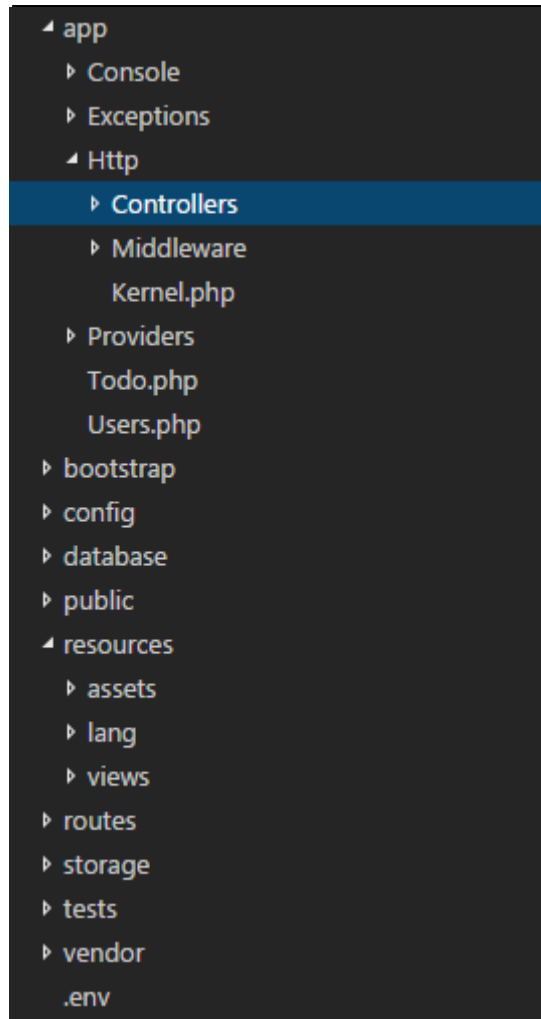
The second method of installation is through Composer **create-project** command.

In the root directory, create a new folder and name it **Laravel**. Now on the command line, go to the **Laravel** folder and use the following code to install Laravel.

composer create-project --prefer-dist Laravel/Laravel <project name>

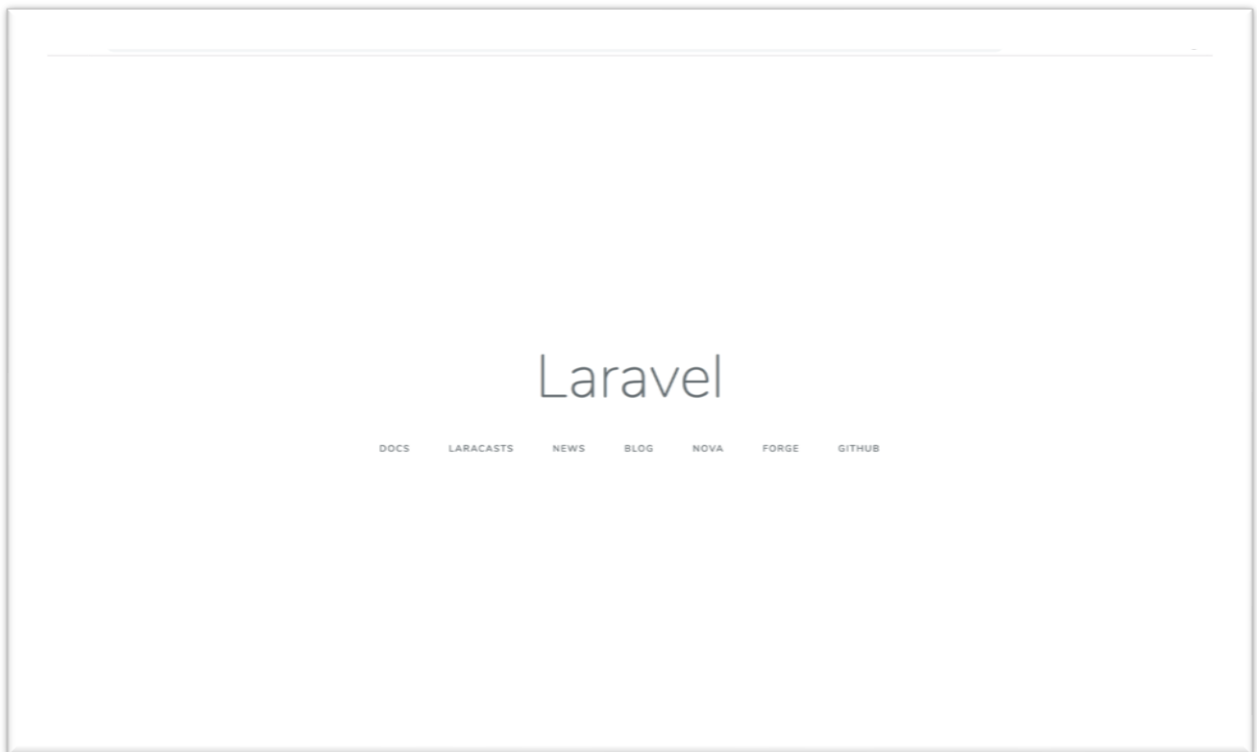
```
[ahmed92]:laravel5.4$ composer create-project --prefer-dist laravel/laravel blog
Installing laravel/laravel (v5.4.0)
- Installing laravel/laravel (v5.4.0) Downloading: 100%
Created project in blog
> php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 59 installs, 0 updates, 0 removals
- Installing symfony/polyfill-mbstring (v1.3.0) Loading from cache
- Installing symfony/var-dumper (v3.2.2) Downloading: 100%
- Installing jakub-ondarka/php-console-color (0.1) Loading from cache
- Installing jakub-ondarka/php-console-highlighter (v0.3.2) Loading from cache
- Installing dnoegel/php-xdg-base-dir (0.1) Loading from cache
- Installing nikic/php-parser (v3.0.2) Loading from cache
- Installing psr/log (1.0.2) Loading from cache
- Installing symfony/debug (v3.2.2) Downloading: 100%
- Installing symfony/console (v3.2.2) Downloading: 100%
- Installing psy/psysh (v0.8.1) Downloading: 100%
- Installing vlucas/phpdotenv (v2.4.0) Loading from cache
- Installing symfony/css-selector (v3.2.2) Downloading: 100%
- Installing tijsverkoyen/css-to-inline-styles (2.2.0) Downloading: 100%
- Installing symfony/routing (v3.2.2) Downloading: 100%
```

Now a new Laravel project has been created. Navigate to the new install directory. The folder structure for a Laravel project is shown below.



The app folder in the Laravel contains the Models and Controllers of the application. Models are created in the root of the app folder, whereas Controllers and Middleware's are created in their respective folders inside the Http folder. Views in Laravel are created in views folder inside the resources folder. Routing for controllers is handled by the Web.php file located inside the routes folder.

At this point, you have a working installation of Laravel on the local host. To test the integrity of the installation, launch the browser and go to **localhost/<project name>/public**.



LARAVEL ROUTING AND HOW LARVEL ROUTING WORKS

Routing is one of the essential and can consider as the core components of the Laravel framework, in In Laravel, all requests are outlined with the guidance of routes. All Laravel routes are determined in the file located as the `app/Http/routes.php` file, which is automatically stored and loaded by the framework. Primary routing named as basic routing routes the call may be referred as a request to the associated controllers. This **Laravel routing tutorial** addresses routing in Laravel.

ROUTING IN LARAVEL COMPRISES THE FOLLOWING CATEGORIES.

- Basic Routing
- Route parameters
- Named Routes

BASIC ROUTING

All the application routes registered itself within the `app/routes.php` file. This file shows Laravel for the URIs it should counter to and the associated controller will dispense it a correct call.

A very basic routes are expressed as follows-

```
Route: :get('/', function () {  
    return 'You are welcome to the Laravel Site. ';
```

It accepts two parameters URL and Closure.

These are the most straightforward routes determined as they don't need you to pass to anything other than the corresponding view. You don't require to log in or give any **Laravel routing parameters** in their route's description or definition.

ACCESSING THE REQUEST

To retrieve an occurrence or instance of the current HTTP **Laravel request object** through dependency injection, you need to type-hint the `Illuminate\Http\Request` class on your controller system or method. The service container will automatically insert or inject the incoming request instance.

```
<?php  
namespace App\Http\Controllers;  
use Illuminate\Http\Request;  
class UserController extends Controller  
{  
    public function store(Request $request)  
    {  
        $name = $request->input('name');  
    }  
}
```

ROUTE PARAMETERS

If the input is also getting expected by the controller method from a route parameter, you must list your route parameters following your different dependencies. For instance, if your route is determined like so:

```
Route::put('user/{id}', 'UserController@update');
```

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class UserController extends Controller
{
    public function update(Request $request, $id)
    {
//code
    }
}
```

1.1.3 INTRODUCTION TO LARAVEL CONTROLLERS

Preferably of determining all of your request administration or handling logic as Closures in route files, you may prefer to build this action applying or using Controller classes. Controllers can group associated request handling logic within a single category or class. Controllers are saved or stored in the app/Http/Controllers of **Laravel controller constructor** directory.

In the MVC framework, the character 'C' stands for Controller. It operates as governing traffic connecting Views and Models. In this chapter, you will discover and learn in depth about Controllers in Laravel.

CREATING CONTROLLERS

Initiate the command prompt or terminal based on the working system you are utilizing and type the following command to generate or create a controller applying the Artisan CLI (Command Line Interface).

php artisan make:controller <controller-name> --plain

You can now replace the <controller-name> with the name of your desired controller, will produce a plain constructor as we are transferring or passing the argument — plain. If you don't require to generate a plain constructor, you can neglect the argument. The created constructor can be viewed at app/Http/Controllers.

You can append your custom coding as you will see the basic coding has already been done for you. The generated controller may be called from routes.php by the following syntax. The example given below is a basic controller class. Perceive that the **Laravel controller constructor** elongates the base controller class involved with Laravel. The base class implements a few convenience systems or method such as the middleware method, which can be applied to append middleware to controller operations:

```
Route::get('base URI','controller@method');
```

```
<?php
namespace App\Http\Controllers;
use App\User;
use App\Http\Controllers\Controller;

class UserController extends Controller
{
    public function show($id)
```

```
{  
    return view('user.profile', ['user' => User::findOrFail($id)]);  
}  
}
```

Resource Controllers

Laravel resource routing specifies or assigns the typical “CRUD” routes to a controller including a single line of code. For instance, you may want to build or create a controller that controls all HTTP requests for “photos” handled by your application.

Php artisan make:controller

php artisan make:controller PhotoController –resource

This command as mentioned earlier will help to generate a controller at `app/Http/Controllers/PhotoController.php`. The controller will comprise a method for each of the accessible resource operations.

Ensuing, you can record an original and resourceful route to the controller:

```
Route::resource('photos', 'PhotoController');
```

This particular, as well as single route declaration, produces various routes to manipulate or handle a variety of activities on the resource.

1.2 MICROSOFT ASP .NET



ASP.NET | MVC | Web API

1.2.1 ASP.NET MVC – OVERVIEW

ASP.NET MVC is basically a web development framework from Microsoft, which combines the features of MVC (Model-View-Controller) architecture, the most up-to-date ideas and techniques from Agile development, and the best parts of the existing ASP.NET platform. ASP.NET MVC is not something, which is built from ground zero. It is a complete alternative to traditional ASP.NET Web Forms. It is built on the top of ASP.NET, so developers enjoy almost all the ASP.NET features while building the MVC application.

Microsoft decided to create their own MVC framework for building web applications. The MVC framework simply builds on top of ASP.NET. Another design goal for ASP.NET MVC was to be extensible throughout all aspects of the framework. The whole idea behind using the Model View Controller design pattern is that you maintain a separation of concerns.

Benefits of ASP.NET MVC

Following are the benefits of using ASP.NET MVC:

- Makes it easier to manage complexity by dividing an application into the model, the
- view, and the controller.
- Enables full control over the rendered HTML and provides a clean separation of
- concerns.
- Direct control over HTML also means better accessibility for implementing compliance
- with evolving Web standards.
- Facilitates adding more interactivity and responsiveness to existing apps.
- Provides better support for test-driven development (TDD).
- Works well for Web applications that are supported by large teams of developers and
- for Web designers who need a high degree of control over the application behavior.

1.2.2 ASP.NET MVC – MVC PATTERN

The MVC architectural pattern separates the user interface (UI) of an application into three main parts

Models

Model objects are parts of the application which implement the logic for the application's data domain. It retrieves and stores model state in a database. For example, product object might retrieve information from a database, operate on it. Then write information back to products table in the SQL server.

Views

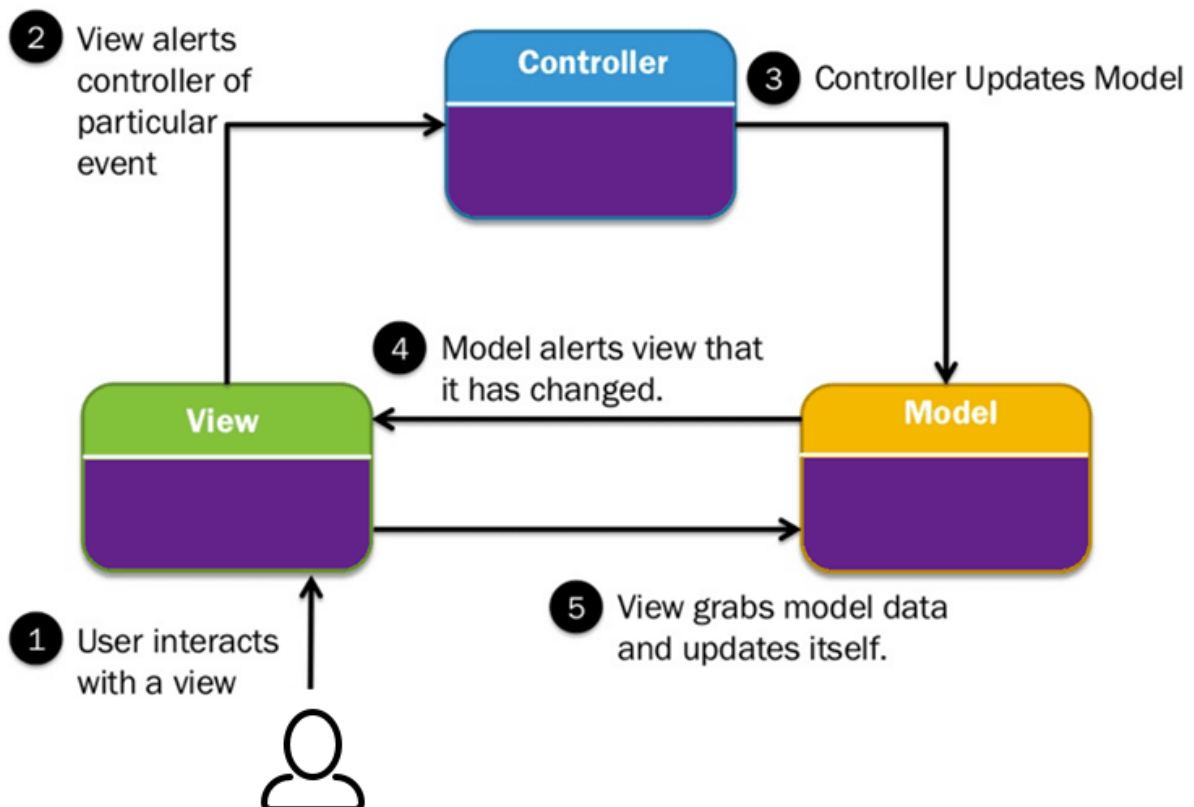
View are the components which are used to display the application's user interface (UI). It displays the .Net MVC application's which is created from the model data.

The common example would be an edit view of an Item table. It displays text

boxes, pop-ups and checks boxes based on the current state of products & object.

Controller

Controllers handle user interaction, work with the model, and select a view to render that displays UI. In a .Net MVC app, the view only displays information, the controller manages and responds to user input & interaction.



The idea is that you'll have a component called the view, which is solely responsible for rendering this user interface. The view talks to a model, and that model contains all of the data that the view needs to display. Views generally don't have much logic inside of them at all. The controller that organizes is everything. When an HTTP request arrives for an MVC application, that request gets routed to a controller, and then it's up to the controller to talk to either the database, the file system, or the model.

ActiveX server pages

For that Microsoft produce first answer as ActiveX server pages (ASP) in this all

code was written on same page (scripts and server code which became hard to understand and Maintain) after that to overcome these things Microsoft produce another solution as Asp.net Web Forms.

ASP.NET Web Forms

Asp.net Web Forms was solution for problem of ASP. The Web Forms came with separation of code in this we have separate UI and server code (HTML | SERVER) this became easy to developer to develop application and fast too. This was best Framework of many developers and many websites are been develop from it. But problem with Frame work was that we cannot reuse code because (.aspx.cs) which is tightly coupled with (.aspx) and this also create problem while testing application because we cannot isolate this application because of tight coupling. For that Microsoft came with new flavor for web developing platform was MVC.

1.2.3 ASP.NET MVC – INSTALLATION

MVC development tool is included with Visual Studio 2012 and onwards. It can also be installed on Visual Studio 2010 SP1/Visual Web Developer 2010 Express SP1. If you are using Visual Studio 2010, Install MVC 4 using the Web Platform Installer

<http://www.microsoft.com/web/gallery/install.aspx?appid=MVC4VS2010>

Microsoft provides a free version of Visual Studio, which also contains SQL Server and it can be downloaded from

<https://www.visualstudio.com/en-us/downloads/download-visualstudio-vs.aspx>.

To develop MVC application we need Visual studio 2010 with Service Pack 1 version. If you are using Visual studio higher then Visual studio 2010 SP1 then it will good for development.

for completing the process, we require good internet connection.



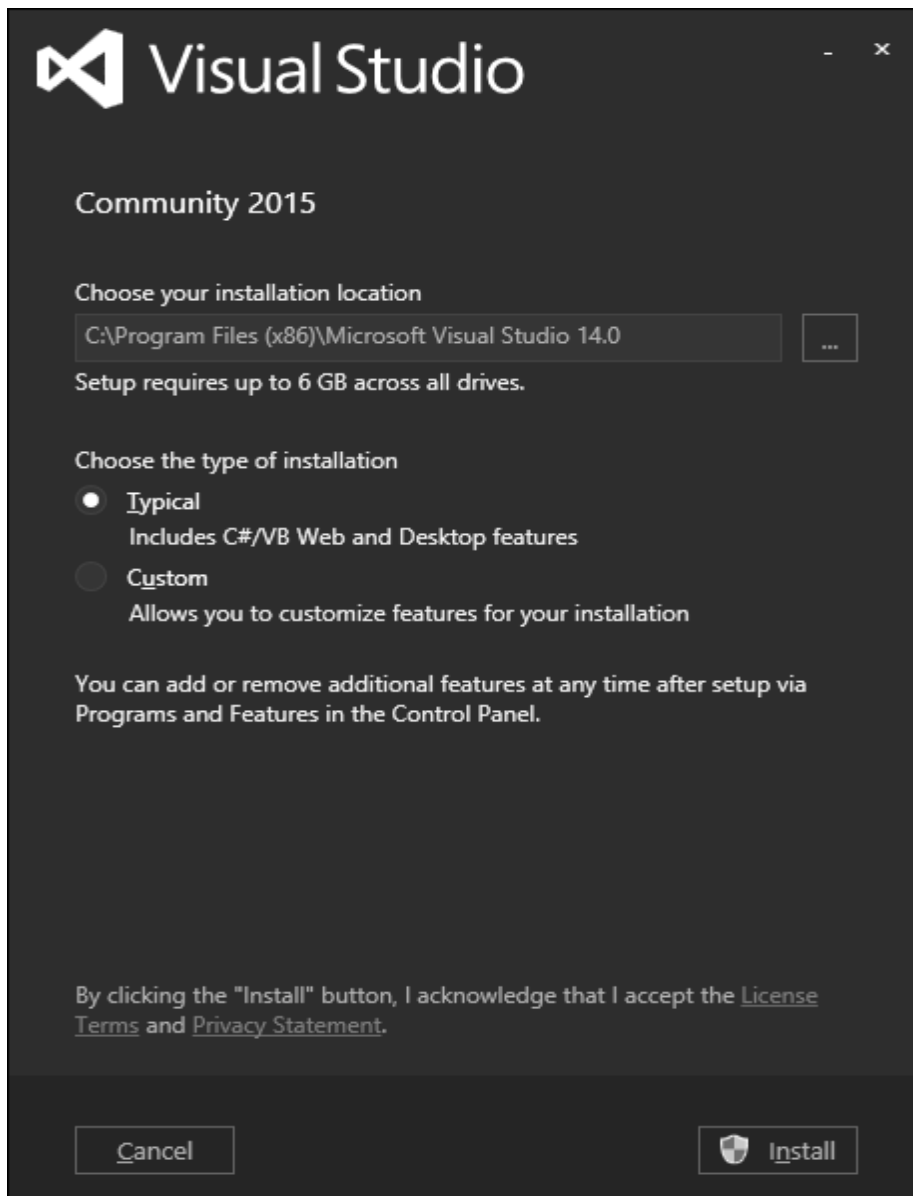
Microsoft Visual Studio 2010 Service Pack 1 (Installer)

Select Language:

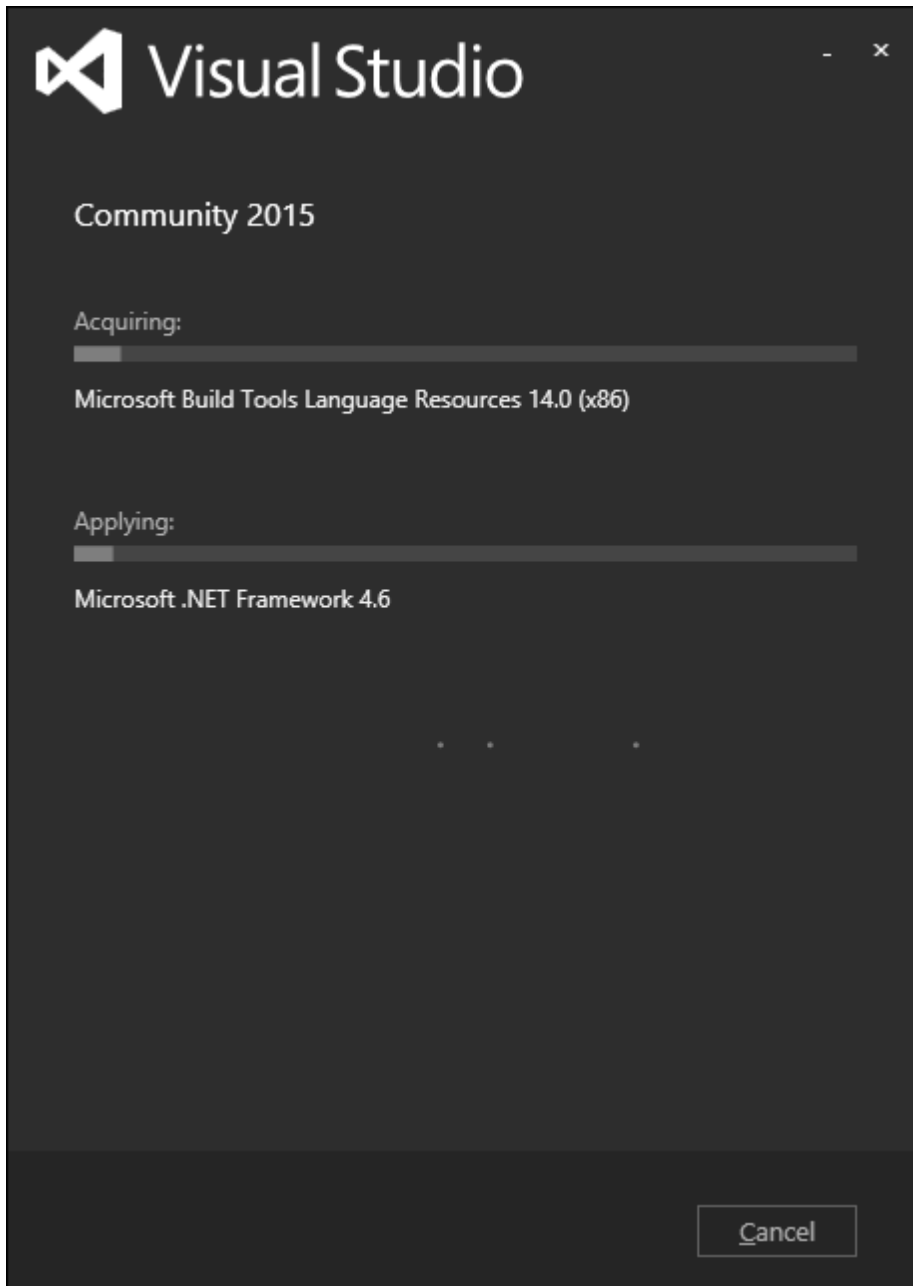
This web installer downloads and installs Visual Studio 2010 Service Pack 1. An Internet connection is required during installation. See the 'Additional Information' section below for alternative (ISO) download options. Please Note: This installer is for all editions of Visual Studio 2010 (Express, Professional, Premium, Ultimate, Test Professional).

Installation

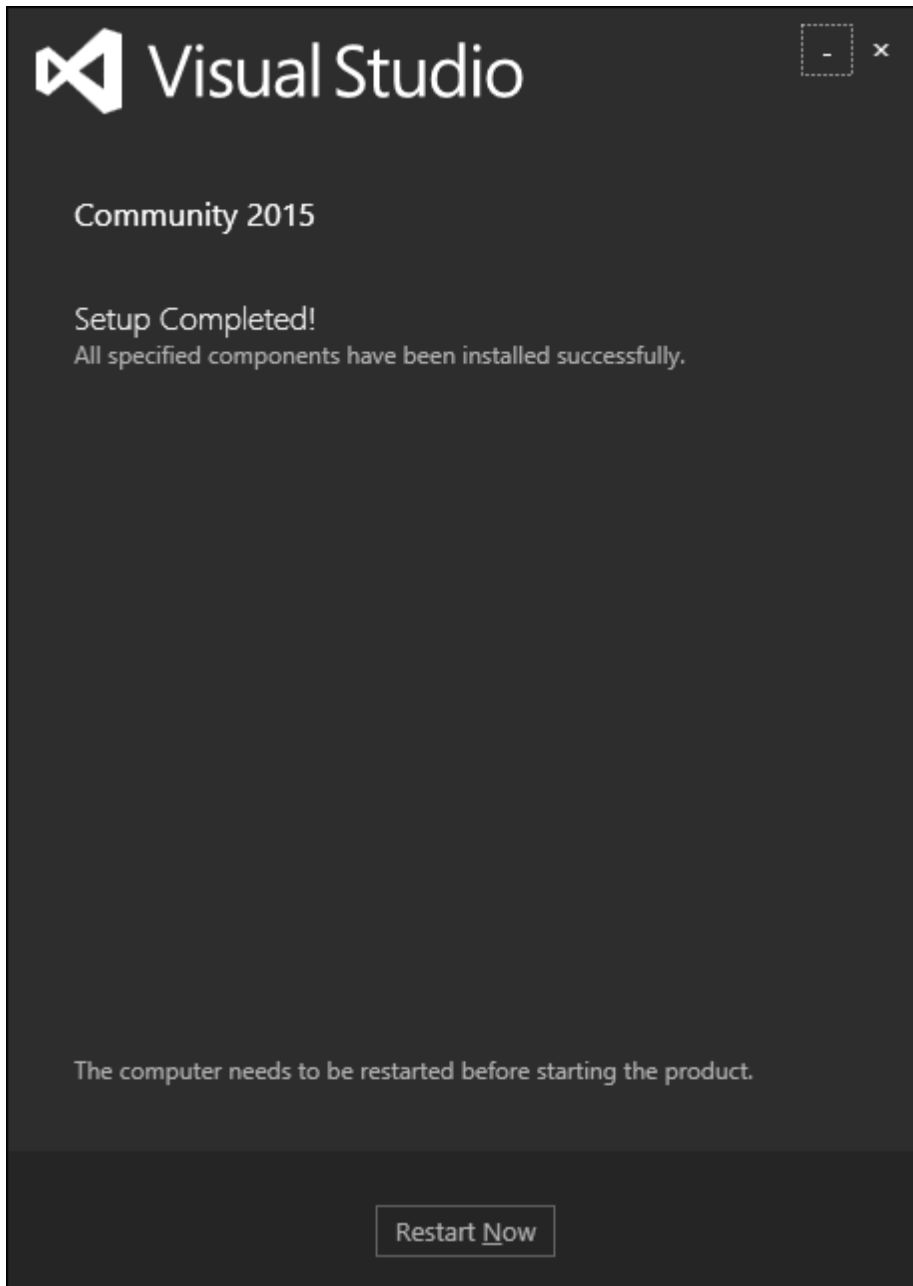
Step (1): Once downloading is complete, run the installer. The following dialog will be displayed.



Step (2): Click the 'Install' button and it will start the installation process.

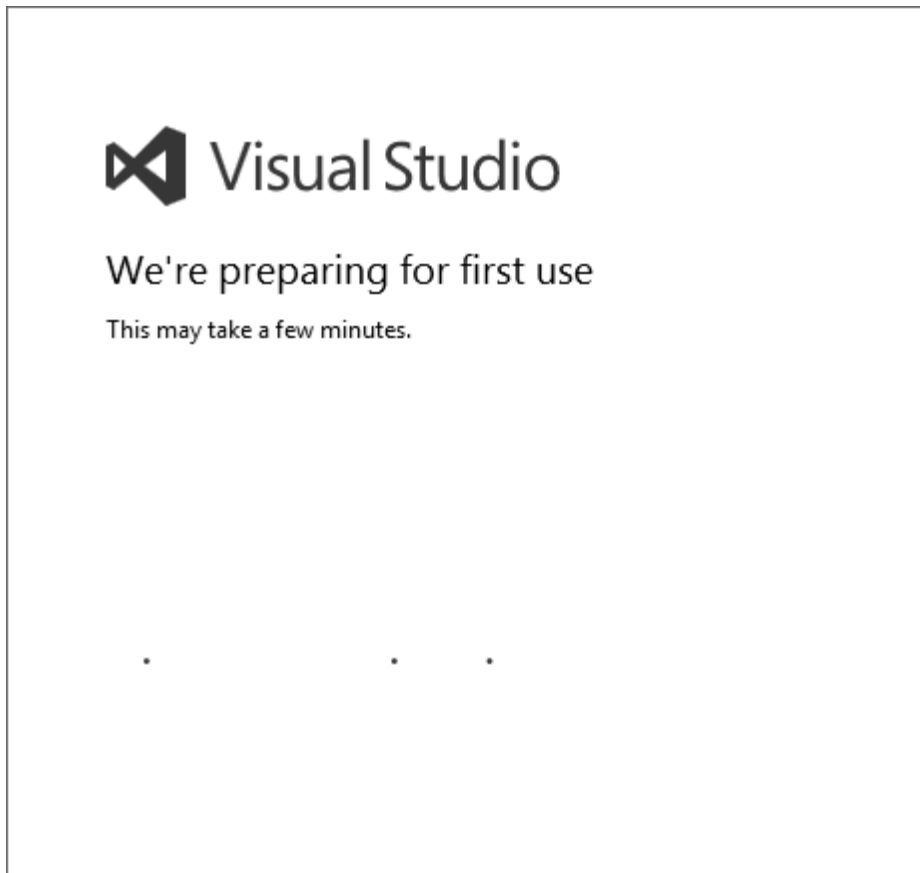


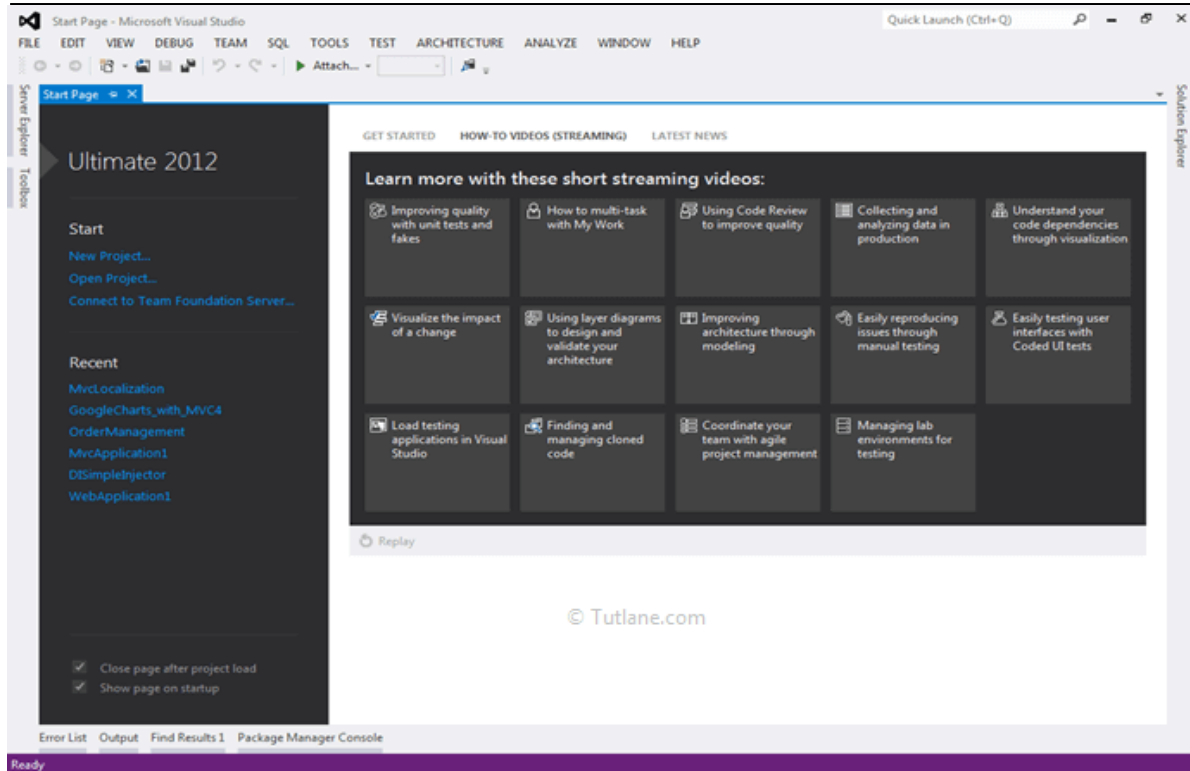
Once the installation process is completed successfully, you will see the following dialog.



Step (3): Close this dialog and restart your computer if required.

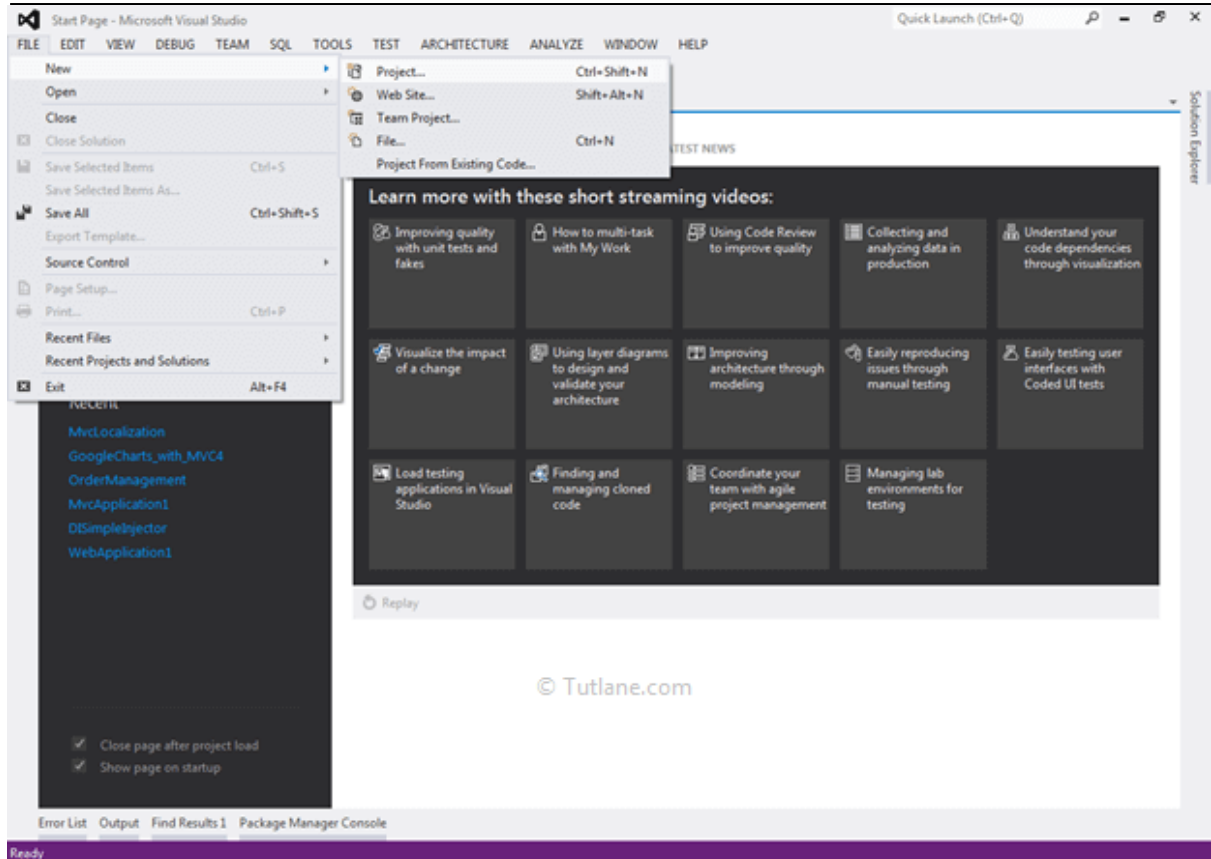
Step (4): Open Visual Studio from the Start Menu, which will open the following dialog. It will take a while for the first time only for preparation.



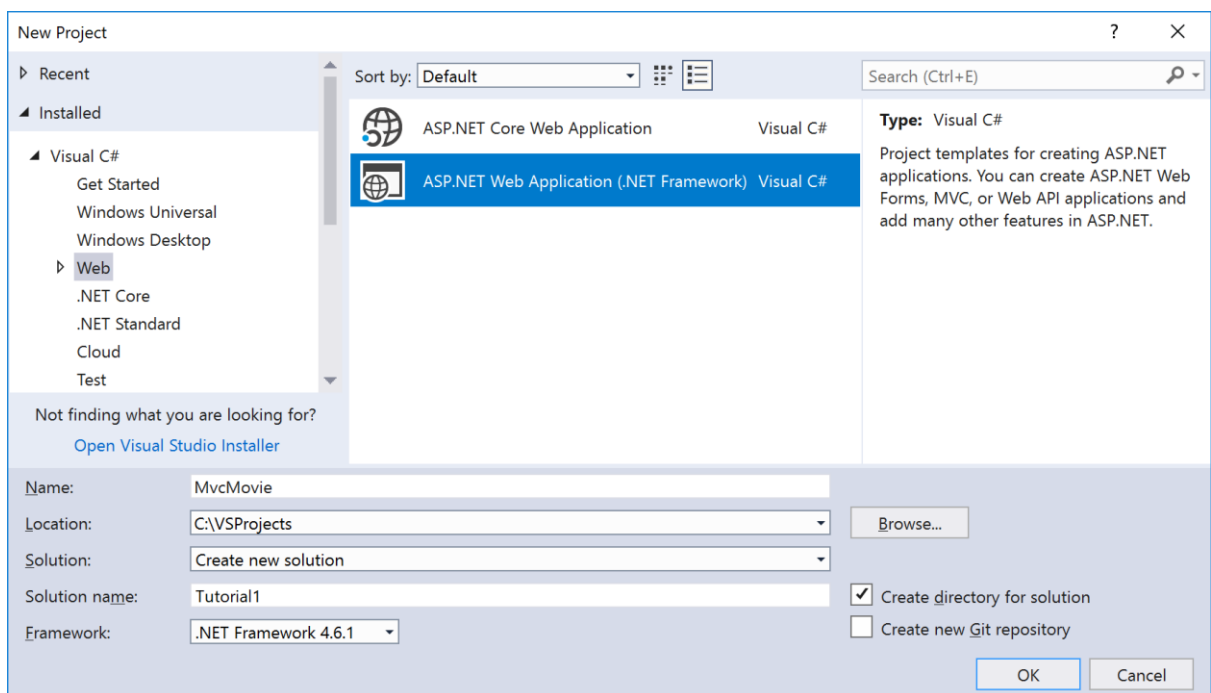


1.2.4 ASP.NET MVC – GETTING STARTED

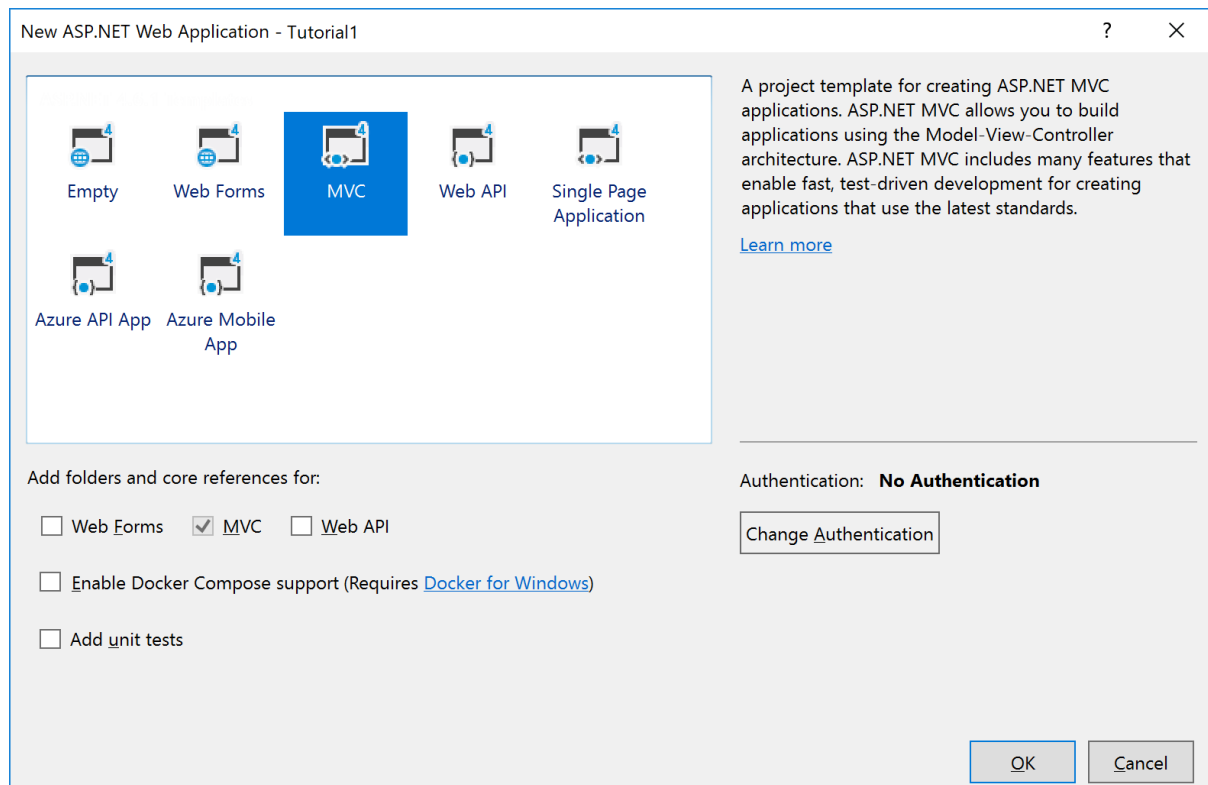
Visual Studio is an IDE, or integrated development environment. Just like you use Microsoft Word to write documents, you'll use an IDE to create applications. In Visual Studio, there's a list along the bottom showing various options available to you. There's also a menu that provides another way to perform tasks in the IDE. For example, instead of selecting **New Project** on the **Start page**, you can use the menu bar and select **File > New Project**.



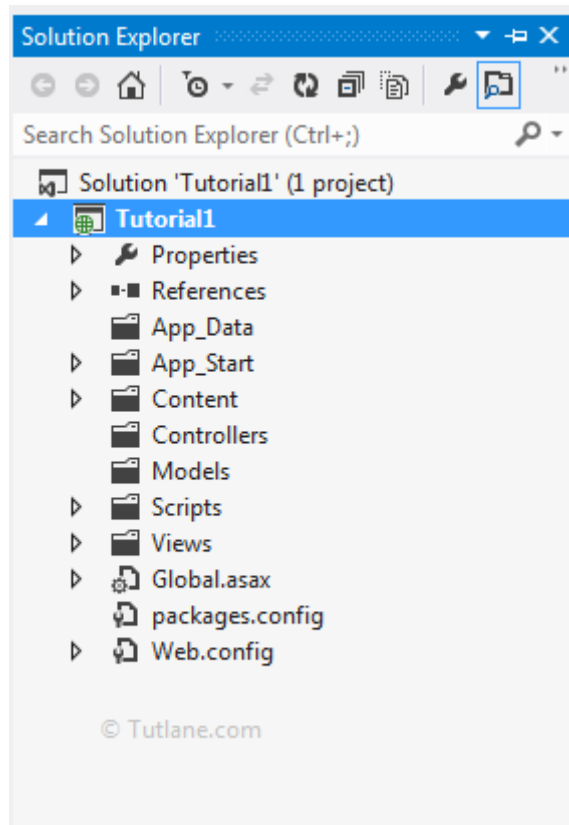
On the **Start** page, select **New Project**. In the **New project** dialog box, select the **Visual C#** category on the left, then **Web**, and then select the **ASP.NET Web Application (.NET Framework)** project template. Name your project and then choose **OK**.



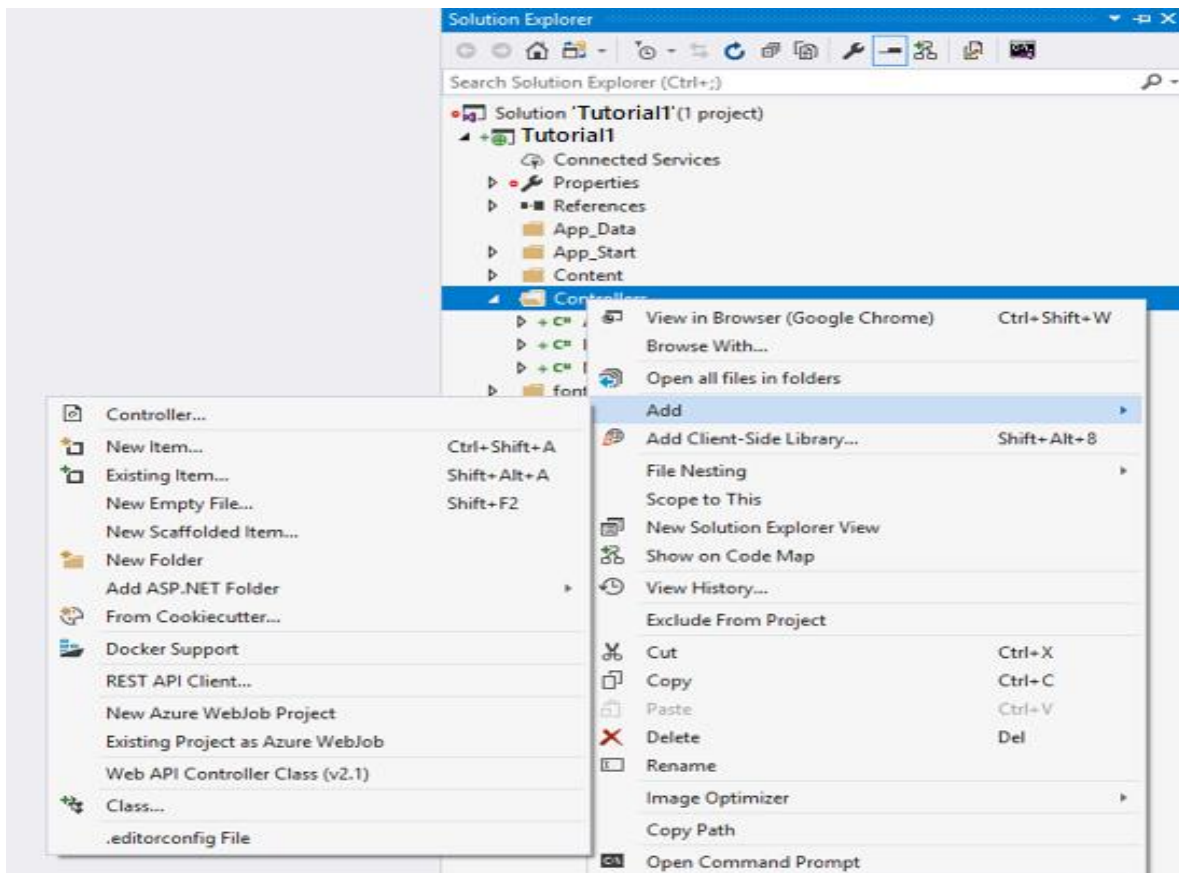
In the **New ASP.NET Web Application** dialog, choose **MVC** and then choose **OK**.



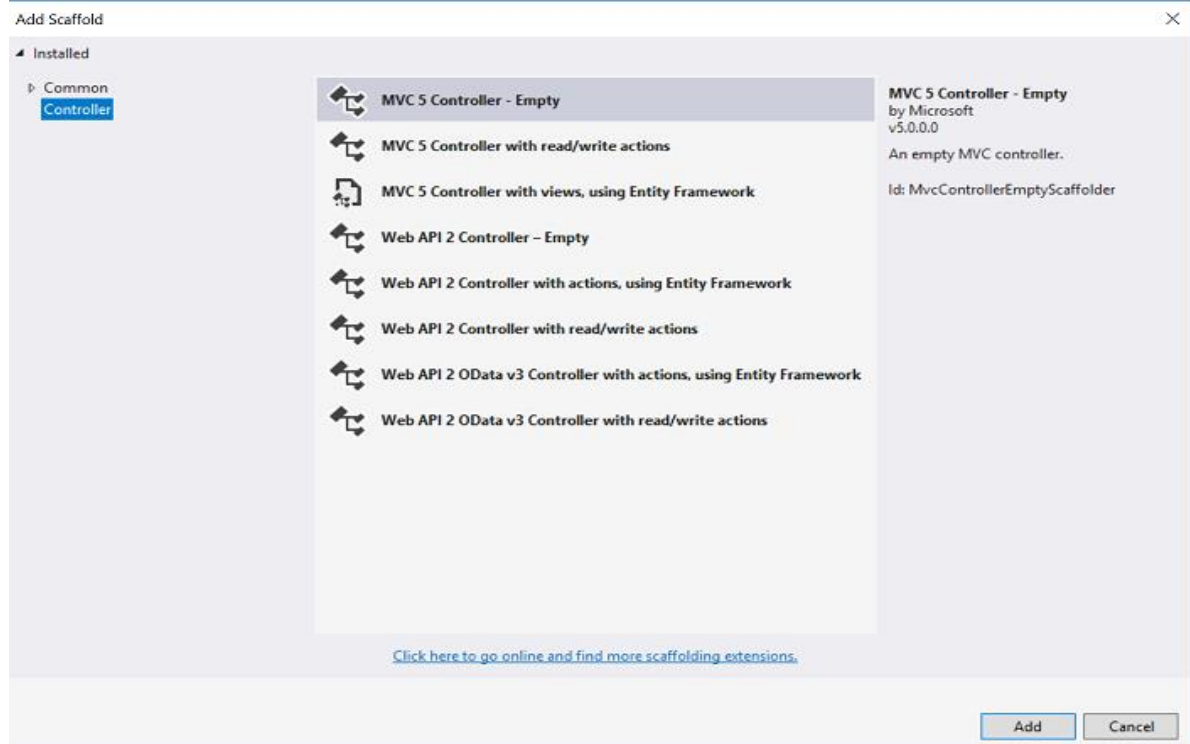
After create application our asp.net mvc application project structure will be like as shown following image.



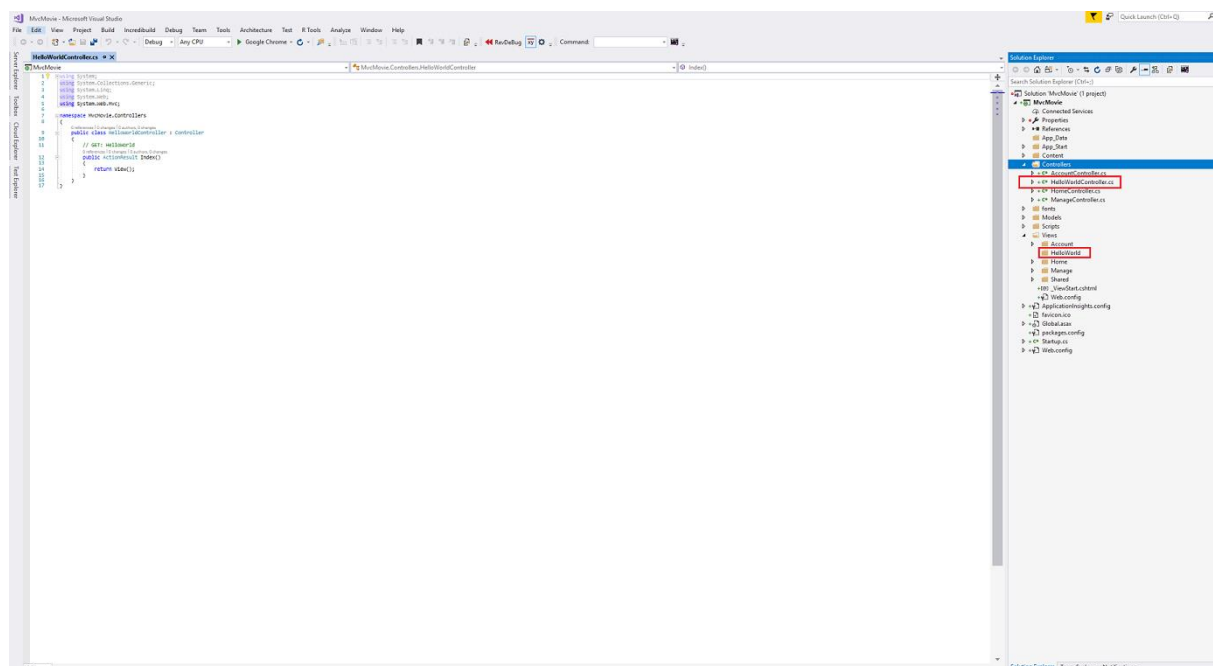
Creating a controller class. In Solution Explorer, right-click the *Controllers* folder and then click Add, then Controller.



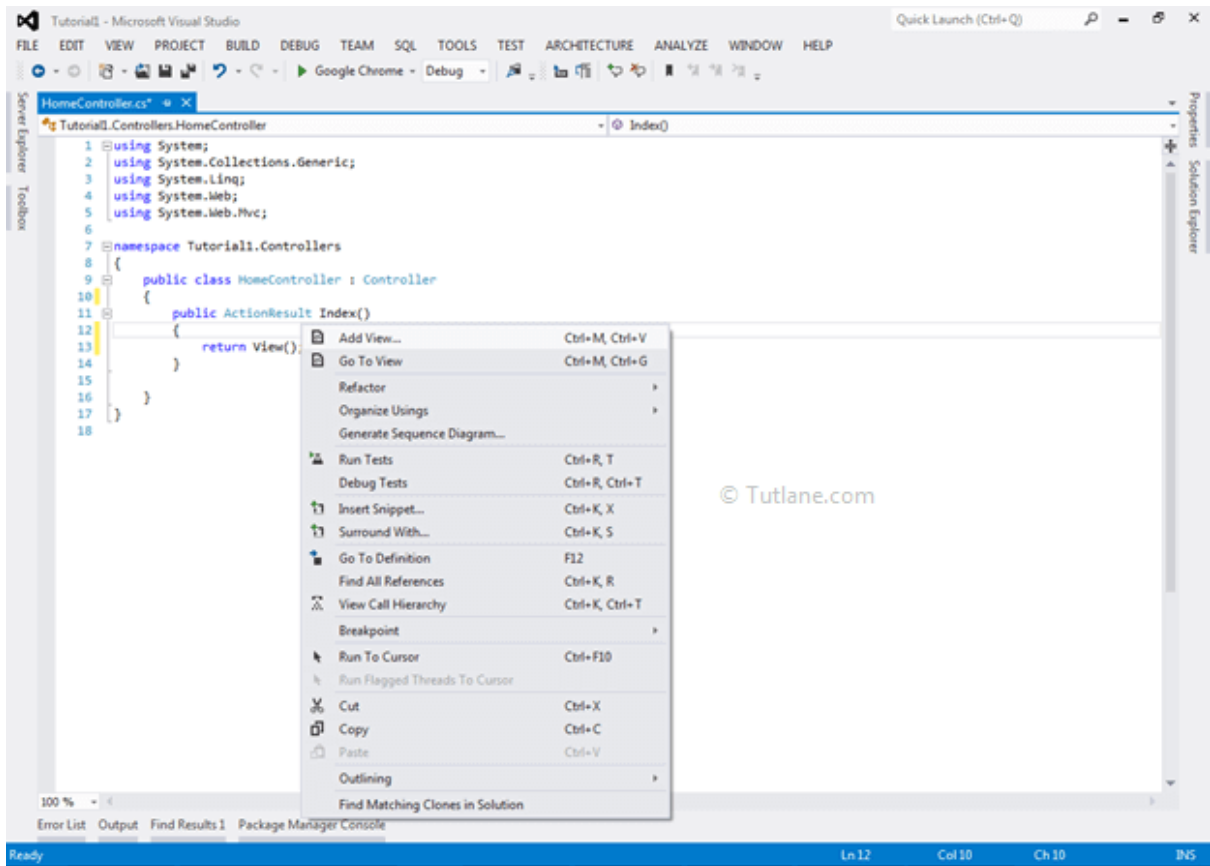
In the **Add Scaffold** dialog box, click **MVC 5 Controller - Empty**, and then click **Add**.



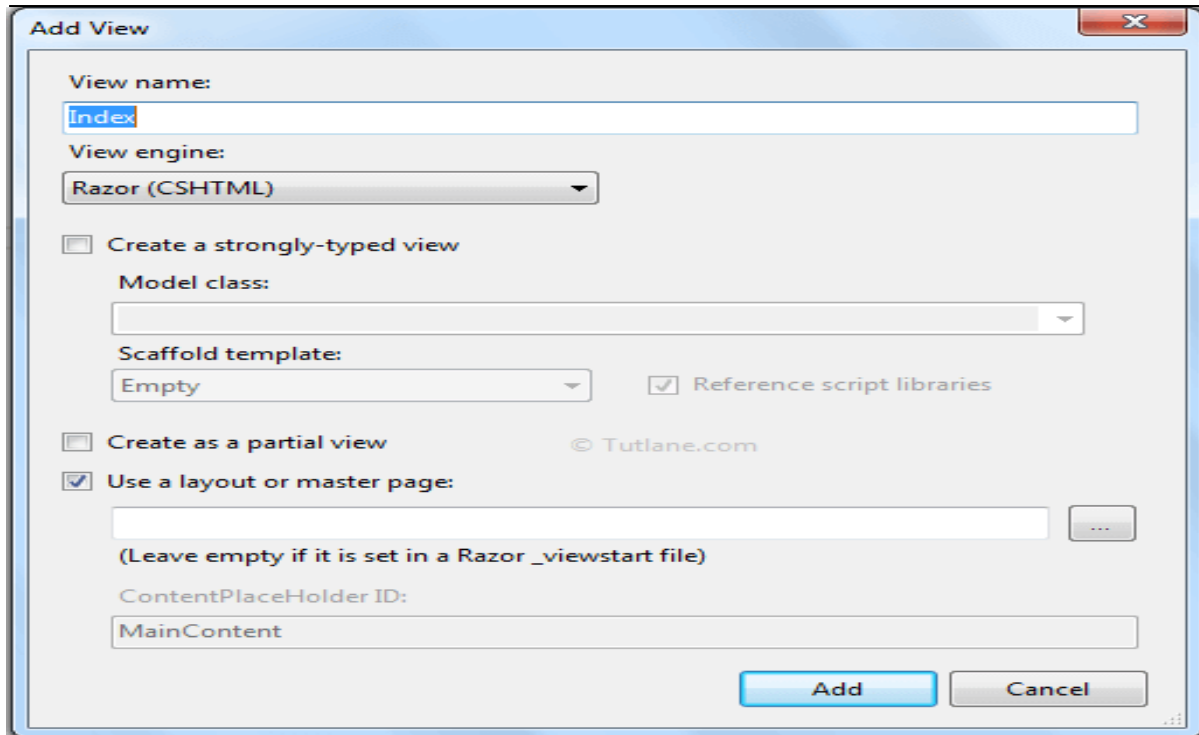
Name your new controller "**HelloWorldController**" and click Add.



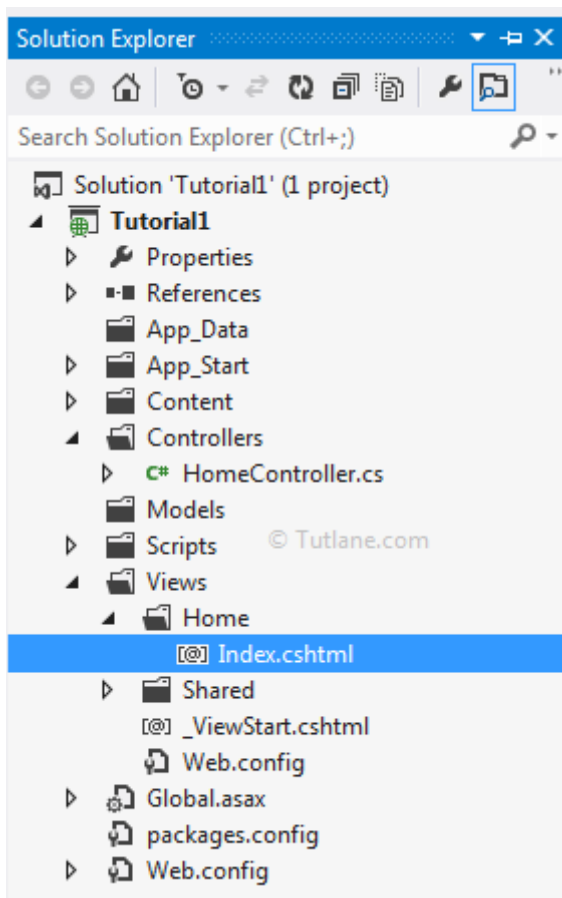
Open **HelloWorldController** file and just right click in inside controller anywhere like as shown following image



After Clicking on Add view a new dialog pop up will open for view configuration like as following image

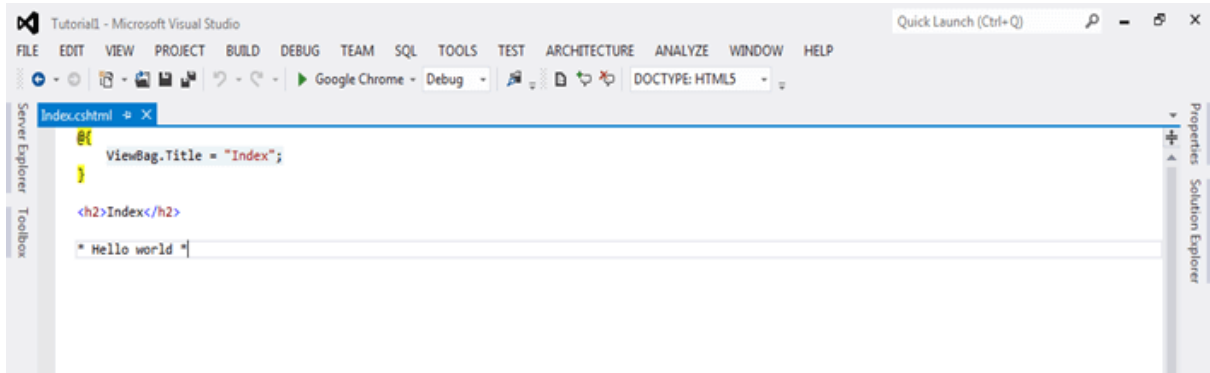


Here we will provide View name as **Index** and click on Add button once we add view our Project structure will be like as shown following image



Now open our Index view and write message "**Hello world**" like as shown

following image



Similarly, this can be done without adding view as:

Code for Hello world program in controller:

using System.Web;

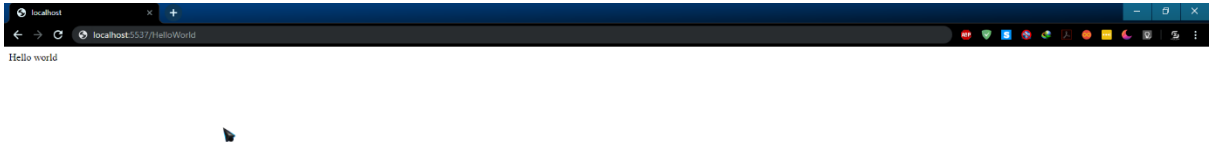
using System.Web.Mvc;

namespace Tutorial1.Controllers

```
{  
    public class HelloWorldController : Controller  
    {  
        public string Index()  
        {  
            return "Hello world";  
        }  
  
        public string Welcome()  
        {  
            return "This is a .net program";  
        }  
    }  
}
```

The controller methods will return a string of HTML as an example. The controller is named **HelloWorldController** and the first method is named Index.

Run the application (press F5 or Ctrl+F5). In the browser, append "HelloWorld" to the path in the address bar.



ASP.NET MVC invokes different controller classes (and different action methods within them) depending on the incoming URL. The default URL routing logic used by ASP.NET MVC uses a format like this to determine what code to invoke:

/[Controller]/[ActionName]/[Parameters]

You set the format for routing in the *App_Start/RouteConfig.cs* file.

```
public static void RegisterRoutes(RouteCollection routes)
{
    routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
    routes.MapRoute(
        name: "Default",
        url: "{controller}/{action}/{id}",
        defaults: new { controller = "Home", action = "Index", id =
        UrlParameter.Optional
    }
}
```

When you run the application and don't supply any URL segments, it defaults to the "Home" controller and the "Index" action method specified in the defaults section of the code above.

The first part of the URL determines the controller class to execute. So */HelloWorld* maps to the HelloWorldController class. The second part of the URL determines the action method on the class to execute. So */HelloWorld/Index* would cause the Index method of the HelloWorldController class to execute. Notice that we only had to browse to */HelloWorld* and the Index method was used by default. This is because a method named Index is the default method that will be called on a controller if one is not explicitly specified. The third part of the URL segment (Parameters) is for route data.

Browse to <http://localhost:xxxx/HelloWorld/Welcome>. The Welcome method runs and returns the string "This is a .NET program". The default MVC mapping is */[Controller]/[ActionName]/[Parameters]*. For this URL, the controller is HelloWorld and Welcome is the action method.



1.3 ANGULAR JS



1.3.1 ANGULAR JS OVERVIEW

AngularJS is a client-side JavaScript MVC framework to develop a dynamic web application. AngularJS was originally started as a project in Google but now, it is open source framework.

AngularJS is entirely based on HTML and JavaScript, so there is no need to learn another syntax or language.

AngularJS changes static HTML to dynamic HTML. It extends the ability of HTML by adding built-in attributes and components and also provides an ability to create custom attributes using simple JavaScript.

Following are the advantages of AngularJS over other JavaScript frameworks:

- Open source JavaScript MVC framework.
- Supported by Google
- No need to learn another scripting language. It's just pure JavaScript and HTML.
- Supports separation of concerns by using MVC design pattern.
- Built-in attributes (directives) makes HTML dynamic.
- Easy to extend and customize.

- Supports Single Page Application.
- Uses Dependency Injection.
- Easy to Unit test.
- REST friendly.

1.3.2 ANGULAR JS INSTALLATION

We need the following tools to setup a development environment for AngularJS:

- AngularJS Library
- Editor/IDE
- Browser
- Web server

Step 1: Install the Angular CLI

You use the Angular CLI to create projects, generate application and library code, and perform a variety of ongoing development tasks such as testing, bundling, and deployment.

Install the Angular CLI globally.

To install the CLI using npm, open a terminal/console window and enter the following command:

npm install -g @angular/cli

Step 2: Create a workspace and initial application

You develop apps in the context of an Angular workspace. A workspace contains the files for one or more projects. A project is the set of files that comprise an app, a library, or end-to-end (e2e) tests.

To create a new workspace and initial app project:

Run the CLI command `ng new` and provide the name `my-app`,

ng new my-app

- The ng new command prompts you for information about features to include in the initial app project. Accept the defaults by pressing the Enter or Return key.
- The Angular CLI installs the necessary Angular npm packages and other dependencies. This can take a few minutes.
- It also creates the following workspace and starter project files:
- A new workspace, with a root folder named my-app
- An initial skeleton app project, also called my-app (in the src subfolder)
- An end-to-end test project (in the e2e subfolder)
- Related configuration files
- The initial app project contains a simple Welcome app, ready to run.

Step 3: Serve the application

Angular includes a server, so that you can easily build and serve your app locally.

- Go to the workspace folder (my-app).
- Launch the server by using the CLI command ng serve, with the --open option.

cd my-app**ng serve --open**

- The ng serve command launches the server, watches your files, and rebuilds the app as you make changes to those files.
- The --open (or just -o) option automatically opens your browser to <http://localhost:4200/>.

Welcome to my-app!



Step 4: Edit Angular component

Components are the fundamental building blocks of Angular applications.

They display data on the screen, listen for user input, and take action based on that input.

As part of the initial app, the CLI created the first Angular component for you.

It is the *root component*, and it is named `app-root`.

- Open `./src/app/app.component.ts`.
- Change the title property from `'my-app'` to `'My First Angular App'`.

src/app/app.component.ts:

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})
```

```
})  
  
export class AppComponent {  
  title = 'My First Angular App!';  
}
```

- The browser reloads automatically with the revised title. That's nice, but it could look better.
- Open ./src/app/app.component.css and give the component some style.

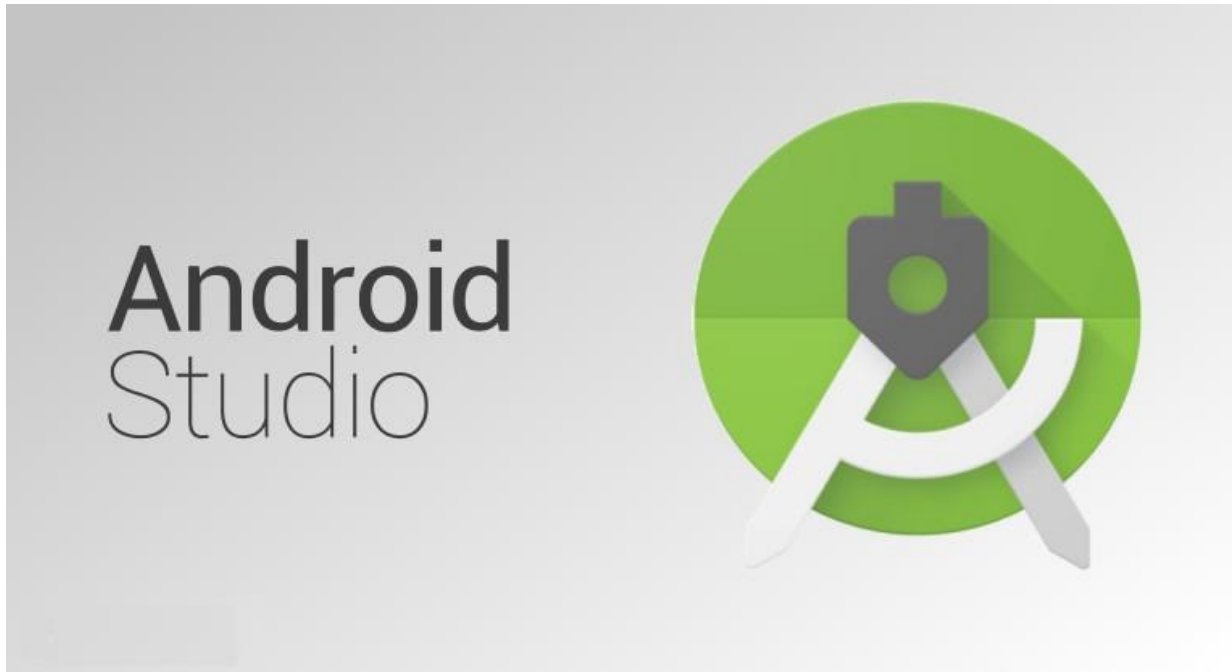
src/app/app.component.css

```
h1 {  
  color: #369;  
  font-family: Arial, Helvetica, sans-serif;  
  font-size: 250%;  
}
```

Welcome to My First Angular App!



1.4 ANDROID



1.4.1 ANDROID IDE OVERVIEW

Android App Development is mostly done in two IDE i.e. Eclipse and Android Studio. Earlier Eclipse was the popular IDE but now Android Studio has taken over it. This is because Google has ended the support for Eclipse and now only focused on Android Studio. Google also recommended developer to import their Android projects and use Android Studio. The *Android Software Development Kit* (Android SDK) and the Gradle tooling contains the necessary tools to Compile, package, deploy and start Android application. Google provides a specialized IDE called *Android Studio* to perform development tasks. The Android SDK contains the *Android debug bridge* (adb). adb is a tool that allows you to connect to a virtual or real Android device. This allows managing the device or debugging your application.

Android is a software package and Linux based operating system for mobile devices such as tablet computers and smartphones. It is developed by Google and later the OHA (Open Handset Alliance). Java language is mainly used to write the android code even though other languages can be used. The goal of android project is to create a successful real-world product that improves the mobile

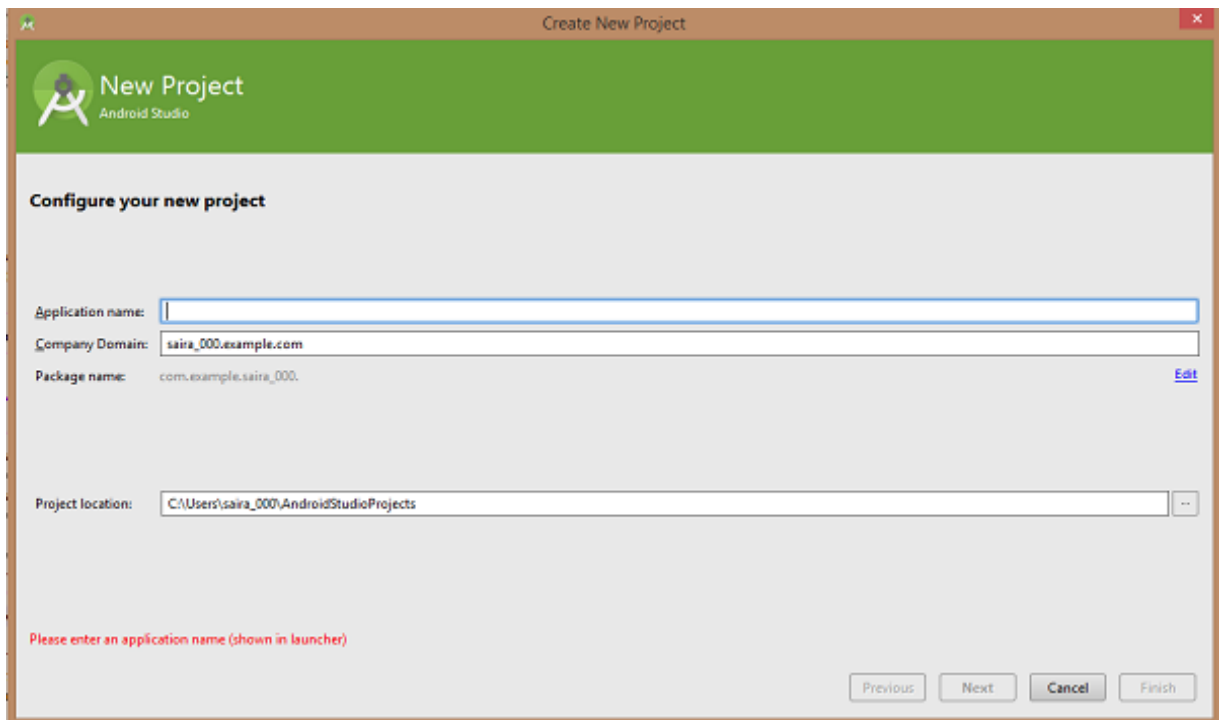
experience for end users. There are many code names of android such as Lollipop, KitKat, Jelly Bean, Ice cream Sandwich, Froyo, Eclair, Donut etc .

1.4.2 Creating Android Application

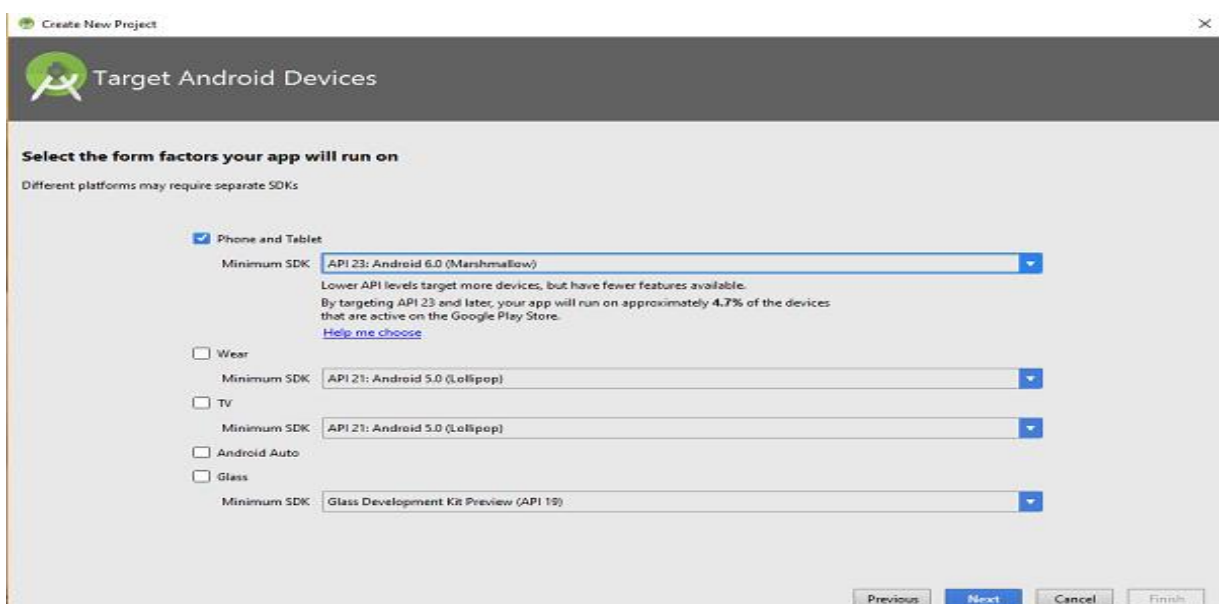
The first step is to create a simple Android Application using Android studio.



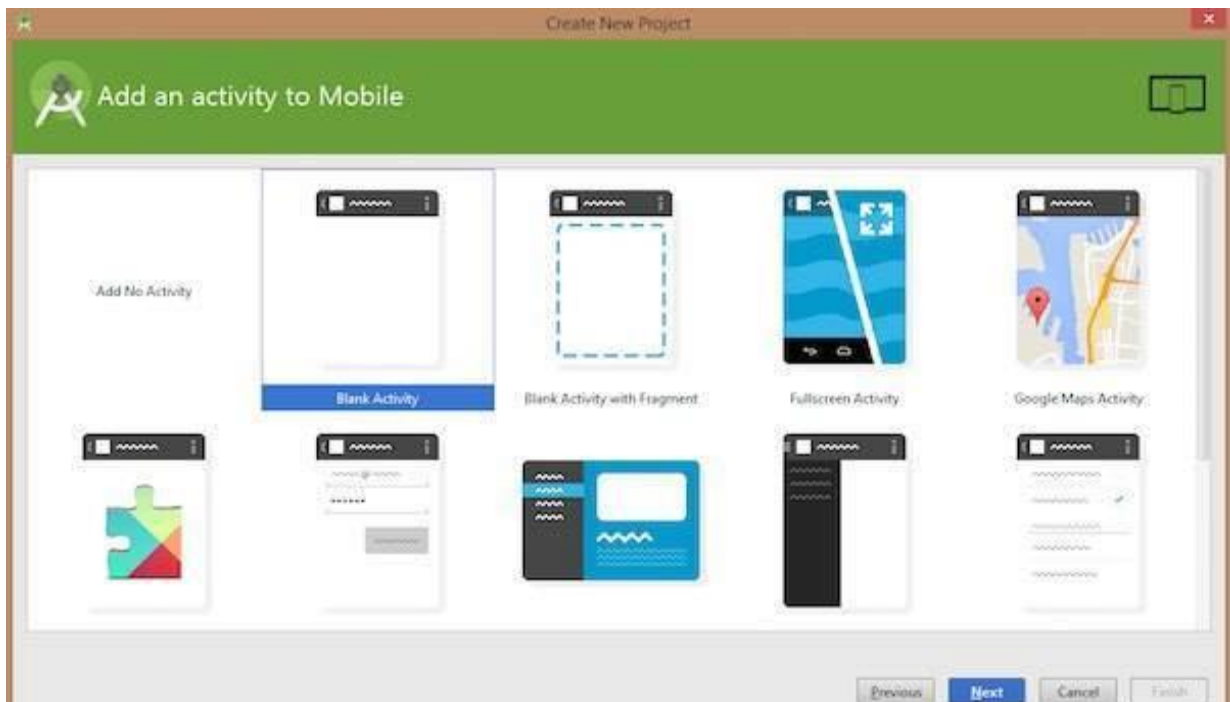
You can start your application development by calling start a new android studio project. in a new installation frame should ask Application name, package information and location of the project.–



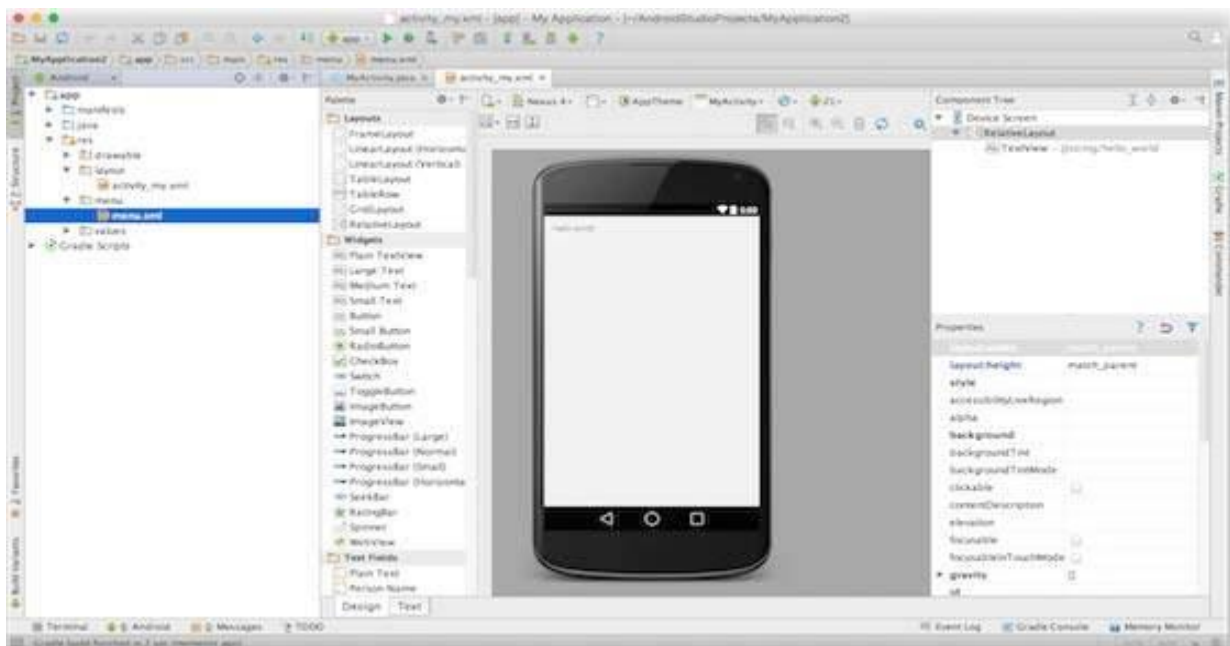
After entered application name, it going to be called select the form factors your application runs on, here need to specify Minimum SDK, in our tutorial, I have declared as API23: Android 6.0(Marshmallow) –



The next level of installation should contain selecting the activity to mobile, it specifies the default layout for Applications.

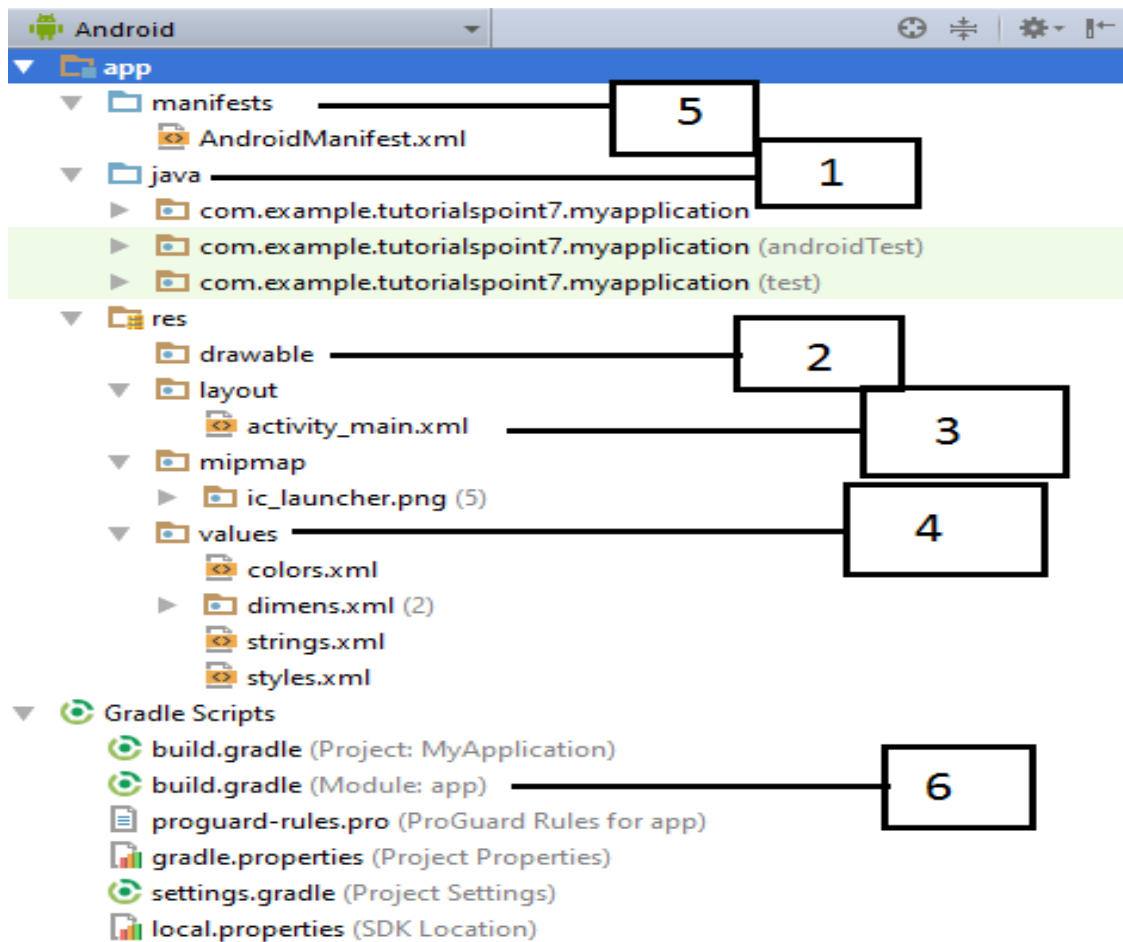


At the final stage it going to be open development tool to write the application code.



1.4.3 ANATOMY OF ANDROID APPLICATION

Before you run your app, you should be aware of a few directories and files in the Android project –



Sr.No.	Folder, File & Description
1	Java This contains the .java source files for your project. By default, it includes an <i>MainActivity.java</i> source file having an activity class that runs when your app is launched using the app icon.
2	res/drawable-hdpi

	This is a directory for drawable objects that are designed for high-density screens.
3	res/layout This is a directory for files that define your app's user interface.
4	res/values This is a directory for other various XML files that contain a collection of resources, such as strings and colours definitions.
5	AndroidManifest.xml This is the manifest file which describes the fundamental characteristics of the app and defines each of its components.
6	Build.gradle This is an auto generated file which contains compileSdkVersion, buildToolsVersion, applicationId, minSdkVersion, targetSdkVersion, versionCode and versionName

Following section will give a brief overview of the important application files.

The Main Activity File

The main activity code is a Java file **MainActivity.java**. This is the actual application file which ultimately gets converted to a Dalvik executable and runs your application. Following is the default code generated by the application wizard for *Hello World!* application –

```
package com.example.helloworld;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

R.layout.activity_main refers to the *activity_main.xml* file located in the *res/layout* folder. The *onCreate()* method is one of many methods that are figured when an activity is loaded.

The Manifest File

Whatever component you develop as a part of your application, you must declare all its components in a *manifest.xml* which resides at the root of the application project directory. This file works as an interface between Android OS and your application, so if you do not declare your component in this file, then it will not be considered by the OS. For example, a default manifest file will look like as following file –

```
<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.tutorialspoint7.myapplication">
```

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">

    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
</manifest>
```

Here `<application>...</application>` tags enclosed the components related to the application. Attribute *android:icon* will point to the application icon available under *res/drawable-hdpi*. The application uses the image named *ic_launcher.png* located in the drawable folders

The `<activity>` tag is used to specify an activity and *android:name* attribute specifies the fully qualified class name of the *Activity* subclass and the *android:label* attributes specifies a string to use as the label for the activity. You can specify multiple activities using `<activity>` tags.

The **action** for the intent filter is named *android.intent.action.MAIN* to indicate that this activity serves as the entry point for the application. The **category** for the intent-filter is named *android.intent.category.LAUNCHER* to indicate that the application can be launched from the device's launcher icon.

The *@string* refers to the *strings.xml* file explained below. Hence, *@string/app_name* refers to the *app_name* string defined in the *strings.xml* file, which is "HelloWorld". Similar way, other strings get populated in the application.

Following is the list of tags which you will use in your manifest file to specify different Android application components –

- <activity>elements for activities
- <service> elements for services
- <receiver> elements for broadcast receivers
- <provider> elements for content providers

The Strings File

The **strings.xml** file is located in the *res/values* folder and it contains all the text that your application uses. For example, the names of buttons, labels, default text, and similar types of strings go into this file. This file is responsible for their textual content. For example, a default strings file will look like following–

```
<resources>
    <string name="app_name">HelloWorld</string>
    <string name="hello_world">Hello world!</string>
    <string name="menu_settings">Settings</string>
    <string name="title_activity_main">MainActivity</string>
</resources>
```

The Layout File

The **activity_main.xml** is a layout file available in *res/layout* directory, that is referenced by your application when building its interface. You will modify this file very frequently to change the layout of your application. For your "Hello World!" application, this file will have following content related to default layout


–

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:padding="@dimen/padding_medium"
        android:text="@string/hello_world"
        tools:context=".MainActivity" />
</RelativeLayout>
```

The *TextView* is an Android control used to build the GUI and it have various attributes like *android:layout_width*, *android:layout_height* etc which are being used to set its width and height etc.. The *@string* refers to the strings.xml file located in the res/values folder. Hence, *@string/hello_world* refers to the hello string defined in the strings.xml file, which is "Hello World!".

Running the Application

Let's try to run our **Hello World!** application we just created. I assume you had created your **AVD** while doing environment set-up. To run the app from Android studio, open one of your project's activity files and click Run  icon from the tool bar. Android studio installs the app on your AVD and starts it and if everything is

fine with your set-up and application, it will display following Emulator window

–



1.5 SERVER HARDENING



1.5.1 SERVER HARDENING OVERVIEW

Server Hardening is the process of enhancing server security through a variety of means which results in a much more secure server operating environment. This is due to the advanced security measures that are put in place during the server hardening process.

The term "hardening," in the general sense, implies taking a soft surface or material and making changes to it which result in that surface becoming stronger and more resistant to damage. That is exactly how **server hardening** impacts server security. Hardened servers are more resistant to security issues than non-hardened servers.

In a time when nearly every computing resource is online and susceptible to attack, server hardening is a near absolute must to perform on your servers. The Internet has vastly altered the complexion of the server hardening industry over the last decade. Much of the applications and system software that is now developed is intended for use on the Internet, and for connections to the Internet. Many servers online today are attacked thousands of times per hour, tens and sometimes hundreds of thousands of times each and every day. The best defense

against such attacks is to ensure that server hardening is a well-established practice within your organization or to outsource this task to an experienced & established server hardening agency.

Server Hardening, probably one of the most important tasks to be handled on your servers, becomes more understandable when you realize all the risks involved. The default config of most operating systems is not designed with security as the primary focus. Instead, default setups focus more on usability, communications and functionality. To protect your servers, you must establish solid and sophisticated server hardening policies for all servers in your organization. Developing a server hardening checklist would likely be a great first step in increasing your server and network security. Make sure that your checklist includes minimum security practices that you expect of your staff. If you go with a consultant you can provide them with your server hardening checklist to use as a baseline.

1.5.2 SERVER HARDENING TIPS & TRICKS

Every server security conscious organization will have their own methods for maintaining adequate system and network security. Often you will find that server hardening consultants can bring your security efforts up a notch with their specialized expertise. Some common server hardening tips & tricks include: - Use Data Encryption for your Communications - Avoid using insecure protocols that send your information or passwords in plain text. - Minimize unnecessary software on your servers. - Disable Unwanted SUID and SGID Binaries - Keep your operating system up to date, especially security patches. - Using security extensions is a plus. - When using Linux, SELinux should be considered. Linux server hardening is a primary focus for the web hosting industry, however in web hosting SELinux is probably not a good option as it often causes issues when the server is used for web hosting purposes. - User Accounts should have very strong passwords - Change passwords

on a regular basis and do not reuse them - Lock accounts after too many login failures. Often these login failures are illegitimate attempts to gain access to your system. - Do not permit empty passwords. - SSH Hardening --- Change the port from default to a non-standard one --- Disable direct root logins. Switch to root from a lower level account only when necessary. - Unnecessary services should be disabled. Disable all instances of IRC - BitchX, bnc, eggdrop, generic-sniffers, guardservices, ircd, psyBNC, ptlink. - Securing /tmp /var/tmp /dev/shm

PART 2

PROJECT DOCUMENTATION

2.1 INTRODUCTION

2.1.1 PROJECT OVERVIEW

Knowhere is a vendor/service discovery initiative in which a user can search and find any type of service available in the location in which the user is on. The application is backed by a user community which provide real user experience reviews and rating for each service or vendor.

With knowhere it's very easy to find outlets and establishments near a place. It also provides information about the offers and services of different vendors from ads. Since it is a community driven application the versatility and availability of information is really good. Users can get genuine responses from the application through the community we have here at knowhere. Outlet vendors can easily list their establishment on knowhere. We've given a hassle-free sign in system. The vendor is required to present valid proof of ownership or particularly any proof that can validate the existence and genuineness of their establishment. Admin at knowhere then validates and verifies the request and vendor is given a confirmation regarding the approval. Then the owner can login to the application and start to polish their postings. Users can use the search facility to find and locate outlets in a location. They are provided with the functionality of rating and review the outlets. If the user doesn't have an account in knowhere then he can enter his email address and verify the same to post a review or comment. Furthermore, users can also find more results for their searches from google maps also.

2.1.2 PROJECT SPECIFICATION

There are mainly three users in the application.

Admin:

- Manager and controller of the entire system.
- Approves Vendors profile
- Complete user management
- Adding categories for vendors listing

Vendors/ Shop owners:

- Each vendor has a profile
- Initially submit necessary details to the admin and admin verifies the gentility of the vendor and approves the profile
- Can add /update/edit profile
- Add description and photos
- Interact with user by answering their queries
- Can display special offers, pricing, Availability, and services as ads

Users:

- Can create an account in the application for requesting online assistance from the vendor. Even though users don't need to have an account for browsing or to review and rate vendors.
- Users without an account is validated by some verification for gaining access to the system (email/Phone verification).
- Review and rate vendors.
- Communicate with vendors.
- Point out or refer missing vendors

2.2 SYSTEM STUDY

2.2.1 INTRODUCTION

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information to recommend improvements on the system. It is a problem-solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studied to the minute's detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the input to the system are identified. The outputs from the organizations are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and decisional variables, analyzing and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action.

A detailed study of the process must be made by various techniques like interviews, questionnaires etc. The data collected by these sources must be scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing system. Now the existing system is subjected to close study and problem areas are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal.

Preliminary study is the process of gathering and interpreting facts, using the

information for further studies on the system. Preliminary study is problem solving activity that requires intensive communication between the system users and system developers. It does various feasibility studies. In these studies, a rough figure of the system activities can be obtained, from which the decision about the strategies to be followed for effective system study and analysis can be take.

2.2.2 PROPOSED SYSTEM

Knowhere is focusing on these key points.

1. Our web application is a platform for both major and minor vendors to announce their service to the public user community. Also, the users can point out or refer the missing vendors which eases the way for identifying vendors.
2. Since our web application is backed by a user community the review and rating feature is quite versatile and widely available.
3. We provide vendors or shop owners with an additional facility that they can advertise on our site regarding their special offers, availability and special services. This can be a way for earning income for the managing admin.
4. Registered users can request appointment, Que status, and custom enquiries or assistance to the vendors. The vendor receives the request as a notification on his mobile or email thus providing a speedy response

ADVANTAGES OF PROPOSED SYSTEM

The system is very simple in design and to implement. The system requires very low system resources and the system will work in almost all configurations. It has got following features:

➤ Better security: -

For data to remain secure measures must be taken to prevent unauthorized access. Security means that data are protected from various forms of destruction. The system security problem can be divided into four related issues: security, integrity,

privacy and confidentiality. Username and password requirement to sign in ensures security. It will also provide data security as we are using the secured databases for maintaining the documents.

➤ *Ensure data accuracy: -*

The proposed system eliminates the manual errors while entering the details of the users during the registration.

➤ *Better service: -*

The product will avoid the burden of hard copy storage. We can also conserve the time and human resources for doing the same task. The data can be maintained for longer period with no loss of data.

➤ *User friendliness and interactive: -*

The proposed system will help the user to reduce the workload and provides user friendly environment so that they can easily do their jobs. The system alerts the users for each activity to be carried out, through notification.

➤ *Minimum time required: -*

The data are management is in such a way that a particular registered user can search service provider very easily.

2.3 REQUIREMENT ANALYSIS

2.3.1 FEASIBILITY STUDY

A feasibility study is carried out to select the best system that meets performance requirements. Feasibility is the determination of whether or not a project is worth doing. The process followed in making this determination is called a feasibility study. This type of study determines if a project can and should be taken. Since the feasibility study may lead to the commitment of large resources, it becomes necessary that it should be conducted competently and that no fundamental errors of judgment are made. Depending on the results of the initial investigation, the survey is expanded to a more detailed feasibility study.

Feasibility study is a test of system proposal according to its workability, impact on the organization, ability to meet user needs, and effective use of resources. The objective of the feasibility study is not to solve the problem but to acquire a sense of its scope. During the study, the problem definition is crystallized and aspects of the problem to be included in the system are determined. All projects are feasible given unlimited resources and infinite time. Unfortunately, the development of computer-based system in many cases is more likely to be plagued by scarcity of resources and delivery date. The document provides the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as Economic, Technical, Resource, Operational and Behavioral feasibilities.

2.3.2 ECONOMIC FEASIBILITY

Economic analysis is the most frequently used method for evaluating the effectiveness of the candidate system. More commonly known as cost/benefit analysis, the procedure is to be determining the benefits and savings that are expected from a candidate and compare them with costs. If benefits outweigh

costs, then the decision is made to design and implement the system. A system's financial benefit must exceed the cost of developing that system. i.e. a new system being developed should be a good investment for the organization. The following are some of the important financial questions asked during preliminary investigation:

- The costs to conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

Explore nearby will be a simple platform for users to access services for their huge needs. It is completely free. Using this system large number people can solve their problems with free of cost. Explore nearby application only needs a basically configured personal computer that has internet connectivity and a browser. So that the system is economically feasible to the users. Being a web application explore nearby will have an associated hosting cost. Since the system doesn't consist of very low multimedia data transfer, so the bandwidth required for the operation of this application is very low. The system will follow the freeware software standards. No cost will be charged from the potential customers. At the initial stage the potential market space will be the local vendors and retail merchant establishments. From these it's clear that the project explore nearby is Economically feasible.

2.3.3 TECHNICAL FEASIBILITY

The system must be evaluated from the technical point of view first. The assessment of this feasibility must be based on an outline design of the system requirement in the terms of input, output, programs and procedures. Having identified an outline system, the investigation must go on to suggest the type of equipment, required method developing the system, of running the system once it has been designed.

Technical issues raised during the investigation were:

- Does the existing technology sufficient for the suggested one?
- Can the system expand if developed?

The project should be developed such that the necessary functions and performance are achieved within the constraints. Through the technology may become obsolete after some period of time, due to the fact that newer version of same software supports older versions, the system may still be used. So, there are minimal constraints involved with this project. The system has been developed using php in front end and MySQL in server in back end, the project is technically feasible for development.

2.3.4 BEHAVIORAL FEASIBILITY

People are inherently resistant to change, and computers have been known to facilitate change. An estimate should be made of how strong a reaction the common user is likely to have toward the development of a new service or vendor discovery application. Therefore, it is understandable that the introduction of a new application requires a lot of efforts to let it reach to the potential users. The software that is being developed is user friendly and easy to learn. In this way, the developed software is truly efficient and can work on any circumstances, tradition, locales.

2.3.5 OPERATIONAL FEASIBILITY

Proposed projects are beneficial only if they can be turned into information systems that will meet the operating requirements of the organization. The test of feasibility asks if the system will work when it is developed and installed. Some of the important questions that are useful to test the operational feasibility of a project are given below Is there sufficient support for the project from the

management? From users? If the present system is well liked and used to the extent that people would not be able to see reasons for a change, there may be a resistance are current methods acceptable to the users? If they are not, users may welcome a change that will bring about a more operational and useful system. Have the users been involved in the planning and development of the project, and then the changes of resistance can be possibly reduced. Issues that appear to be quite minor at the early stage can grow into major problems after implementation.

2.4 REQUIREMENT MODELING

2.4.1 REQUIREMENT ANALYSIS

➤ To what extend the system is proposed for?

Google maps has already implemented something very similar to this. It's called 'Explore nearby' in which a user can search for services available near the user's location. The three key shortcomings for this feature are:

- Only major services are pinned in the map. Restaurants, Banks, Hospitals and such are marked whereas minor or low-level local vendors are missed out in the map.
- In order for a service or vendor to be marked in the map the vendor should care for it. Also, the process for pinning a particular service or vendor to the map is quite complicated. Even though there's a review and rating system, it's not so that efficient.

Our application is focusing on these three key points.

- Our web application is a platform for both major and minor vendors to announce their service to the public user community. Also, the users can point out or refer the missing vendors which eases the way for identifying vendors.

- Since our web application is backed by a user community the review and rating feature is quite versatile and widely available.
- We provide vendors or shop owners with an additional facility that they can advertise on our site regarding their special offers, availability and special services. This can be a way for earning income for the managing admin.
- Registered users can request appointment, Que status, and custom enquiries or assistance to the vendors. The vendor receives the request as a notification on his mobile or email thus providing a speedy response

➤ Specify the Viewers/Public which is to be involved in the System?

The viewers of the application are general public. Any user can search for any vendor in the application without any restrictions. Even though, if the user is not a registered user of the application, he has to verify his identity by google sign in.

➤ List the Modules included in your System?

- **Login/Registration:** Shop owners can register and create account. Users can create account for availing assistance from vendors
- **Refer/request a profile:** Vendors need to submit their basic info to the admin in order for it to be validated and verified. Users can point out or refer missing vendors.
- **Edit profile:** Vendors and users can edit their profile information. Vendors are given more options for providing detailed information regarding their shop to the public.
- **Search:** Users can search for vendors with filtering by category
- **Add category:** Admin creates categories and sub categories and assigns vendors to their respective category.

- **Rate and Review:** Users can rate and review vendors. Reviews and ratings are done only if a verification key provided by the vendor is entered and is validated correct, thus eliminating fake and unnecessary reviews.
- **Payment:** Vendors can advertise on the site. Users can fix appointments or request que status from vendors and pre-order things from the shop

➤ Identify the users in your project?

Admin:

- Manager and controller of the entire system.
- Approves Vendors profile
- Complete user management
- Adding categories for vendors listing

Vendors/ Shop owners:

- Each vendor has a profile
- Initially submit necessary details to the admin and admin verifies the Genuity of the vendor and approves the profile.
- Can add /update/edit profile
- Add description and photos
- Interact with user by answering their queries
- Can display special offers, pricing, Availability, and services as ads

Users:

- Can create an account in the application for requesting online assistance from the vendor. Even though users don't need to have an account for browsing or to review and rate vendors.
- Users without an account is validated by some verification for gaining access to the system (either by google login or email/Phone verification)
- Review and rate vendors
- Communicate with vendors

- Point out or refer missing vendors

➤ Who owns the system?

The managing admin owns the system. Admin has the complete control of the system and he earns the profit income of the application.

➤ System is related to which firm/industry/organization?

The system is related to merchant-customer market. This application enables users to discover vendor services suiting for their needs without visiting each shop of the same type. Also, customers can best know the value of merchant shops by consulting the reviews provided by other users.

➤ Details of person that you have contacted for data collection?

- Mr. Mathew Abraham, tourist taxi driver, Empire tours and travels kattappana

He has been a tourist's taxi driver for 20 years now. He has dealt with several people from different places, been to so many places.

- Mr. Manoj P.V, Bakery shop owner, Calvary mount- Kattappana

He's a local bakery merchant near Calvary mount which is a famous tourist spot in Idukki. He has dealt with a lot of customers from different places.

Questionnaire to collect details about the project? (min 10 questions, include descriptive answers, attach additional docs (e.g. Bill receipts, certificate models), if any?)

Mr. Mathew Manuel-

- You've been to many places and accompanied many people in their journeys. In your experience how hard it is to locate some specific service providing shop in an unknown city or place?

Ans: When we're going to a rural place or a big town, locating a shop or merchant is quite a tedious task. We've to either seek local help or should have prior experience in the place. As of now we mostly use google maps for finding routes, locating places and merchants.

- How's your experience with google maps in locating merchants?

Ans: It's a great help that google maps points merchants. Even though it points major vendors, many shops and services are missing from the map. It's a hard job to find them in an unknown place. We look for reviews and ratings for the place, but they are not good for getting a complete vision about the vendor.

- How do you find such vendors/shops?

Ans: The only way is to enquire local people

- Even if you find some shop for your need do, they satisfy your needs?

Ans: First of all, it's so hard to find these local merchants. Even if we find some, they wouldn't provide the required service we seek for.

- In your opinion, how can we remedy this problem?

Ans: if google map pints such local shops it would have been so good. But I think it's not a feasible solution. Also, by getting detailed and genuine reviews about the shop will be of great help.

Mr. Manoj P.V-

- Since your shop is in a tourist place, have you helped people coming from far places in finding their destination?

Ans: Many people come to my shop daily enquiring for merchants. We give information that we know. These people coming from far places have no knowledge about local places and shops. So, getting genuine information is a hard thing for them.

- Have you had any bitter experience in finding such services anywhere?

Ans: once I went to cochin for attending a marriage function, I had to roam

around for 2 hours to locate a shop for getting a haircut.

- Do you have any suggestion in how can we solve this problem?

Ans: now a days google maps is the help for finding places. But local merchant shops are not pointed in google maps. I don't think they'll be able to accommodate all these local shops in the map. It'll make the map so crowded so that's not happening.

2.4.2 UML USE CASE DIAGRAM

A UML use case diagram is the primary form of system/software requirements for a new software program under developed. Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (such as UML). A key concept of use case modeling is that it helps us design a system from end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior. The purpose of a use case diagram in UML is to demonstrate the different ways that a user might interact with a system.

In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

- Scenarios in which your system or application interacts with people, organizations, or external systems
- Goals that your system or application helps those entities (known as actors) achieve
- The scope of your system

A use case diagram doesn't go into a lot of detail—for example, don't expect it to

model the order in which steps are performed. Instead, a proper use case diagram depicts a high-level overview of the relationship between use cases, actors, and systems. Experts recommend that use case diagrams be used to supplement a more descriptive textual use case.

UML is the modeling toolkit that you can use to build your diagrams. Use cases are represented with a labeled oval shape. Stick figures represent actors in the process, and the actor's participation in the system is modeled with a line between the actor and use case. To depict the system boundary, draw a box around the use case itself. UML use case diagrams are ideal for:

- Representing the goals of system-user interactions
- Defining and organizing functional requirements in a system
- Specifying the context and requirements of a system
- Modeling the basic flow of events in a use case
- Specify the context of a system
- Capture the requirements of a system
- Validate a systems architecture
- Drive implementation and generate test cases
- Developed by analysts together with domain experts

How to Draw a Use Case Diagram?

Use case diagrams are considered for high level requirement analysis of a system. When the requirements of a system are analyzed, the functionalities are captured in use cases. We can say that use cases are nothing but the system functionalities written in an organized manner. The second thing which is relevant to use cases are the actors. Actors can be defined as something that interacts with the system. Actors can be a human user, some internal applications, or may be some external applications. When we are planning to draw a use case diagram, we should have the following items identified.

Functionalities to be represented as use case

Actors

Relationships among the use cases and actors.

Use case diagrams are drawn to capture the functional requirements of a system. After identifying the above items, we have to use the following guidelines to draw an efficient use case diagram. The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed.

- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements.
- Use notes whenever required to clarify some important points.

To understand the dynamics of a system, we need to use different types of diagrams. Use case diagram is one of them and its specific purpose is to gather system requirements and actors. Use case diagrams specify the events of a system and their flows. But use case diagram never describes how they are implemented. Use case diagram can be imagined as a black box where only the input, output, and the function of the black box is known. High-level design is refined again and again to get a complete and practical picture of the system. A well-structured use case also describes the pre-condition, post condition, and exceptions. These extra elements are used to make test cases when performing the testing.

Although use case is not a good candidate for forward and reverse engineering, still they are used in a slightly different way to make forward and reverse engineering. The same is true for reverse engineering. Use case diagram is used differently to make it suitable for reverse engineering. In forward engineering, use case diagrams are used to make test cases and in reverse engineering use cases are used to prepare the requirement details from the existing application.

Basic Use Case Diagram Symbols and Notations

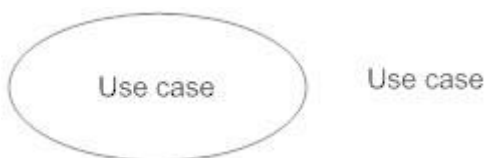
System

Draw your system's boundaries using a rectangle that contains use cases. Place actors outside the system's boundaries.



Use Case

Draw use cases using ovals. Label the ovals with verbs that represent the system's functions.



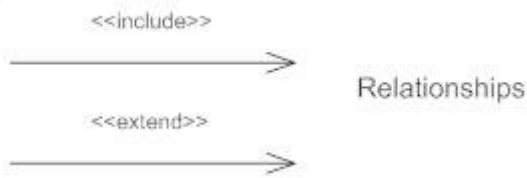
Actors

Actors are the users of a system. When one system is the actor of another system, label the actor system with the actor stereotype.



Relationships

Illustrate relationships between an actor and a use case with a simple line. For relationships among use cases, use arrows labeled either "uses" or "extends." A "uses" relationship indicates that one use case is needed by another in order to perform a task. An "extends" relationship indicates alternative options under a certain use case.



How to Create a Use Case Diagram

- **Identifying Actors**

Actors are external entities that interact with your system. It can be a person, another system or an organization. In a banking system, the most obvious actor is the customer. Other actors can be bank employee or cashier depending on the role you're trying to show in the use case.

- **Identifying Use Cases**

A good way to do this is to identify what the actors need from the system. In a banking system, a customer will need to open accounts, deposit and withdraw funds, request check books and similar functions. So, all of these can be considered as use cases. Top level use cases should always provide a complete function required by an actor. You can extend or include use cases depending on the complexity of the system. Once you identify the actors and the top-level use case you have a basic idea of the system. Now you can fine tune it and add extra layers of detail to it.

- **Look for Common Functionality to use Include**

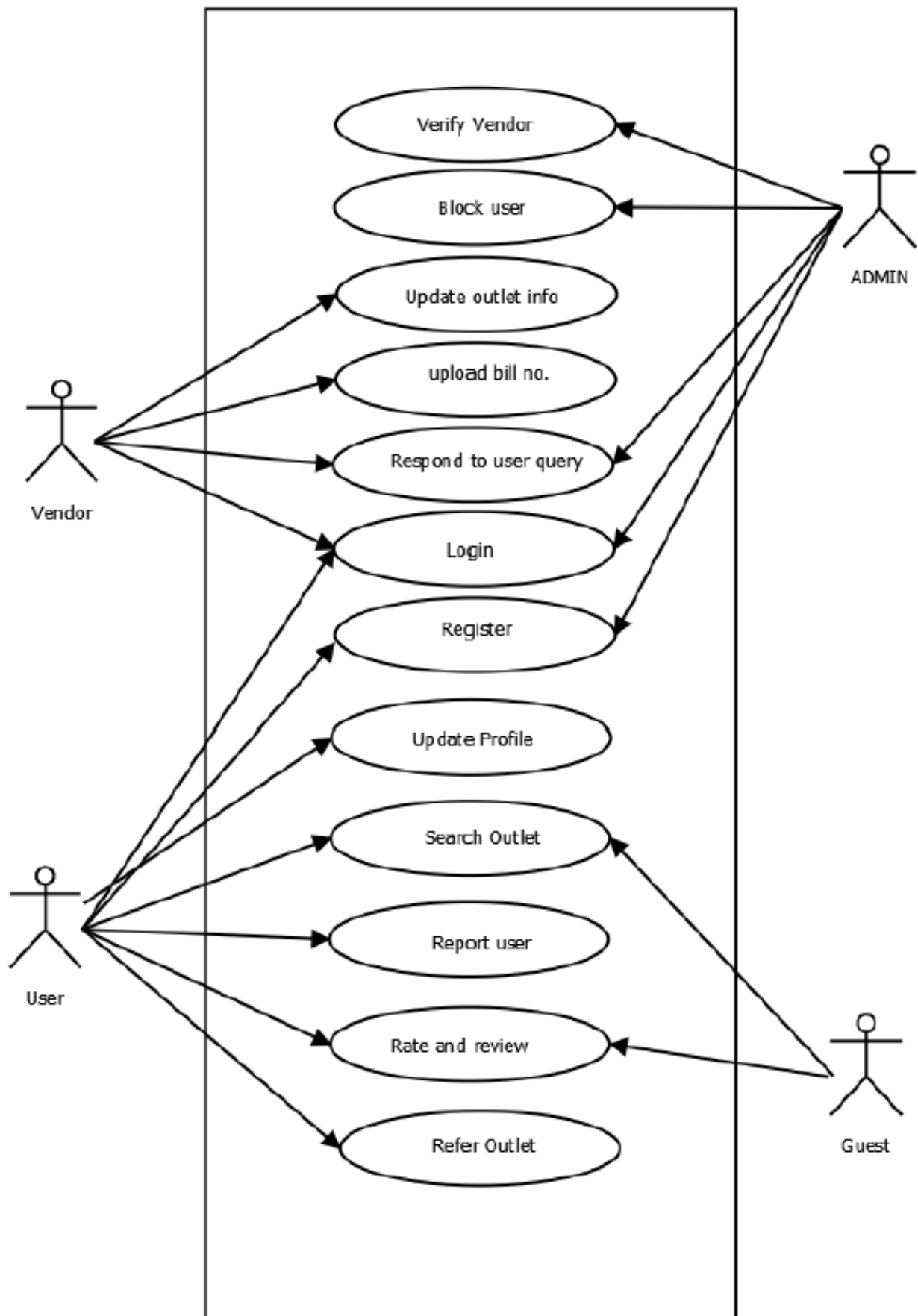
Look for common functionality that can be reused across the system. If you find two or more use cases that share common functionality you can extract the common functions and add it to a separate use case. Then you can connect it via the include relationship to show that it's always called when the original use case is executed.

- **Is it Possible to Generalize Actors and Use Cases?**

There may be instances where actors are associated with similar use cases while triggering few use cases unique only to them. In such instances, you can generalize the actor to show the inheritance of functions. You can do a similar thing for use case as well.

- **Optional Functions or Additional Functions**

There are some functions that are triggered optionally. In such cases, you can use the extend relationship and attach an extension rule to it. In the below banking system example “Calculate Bonus” is optional and only triggers when a certain condition is matched.



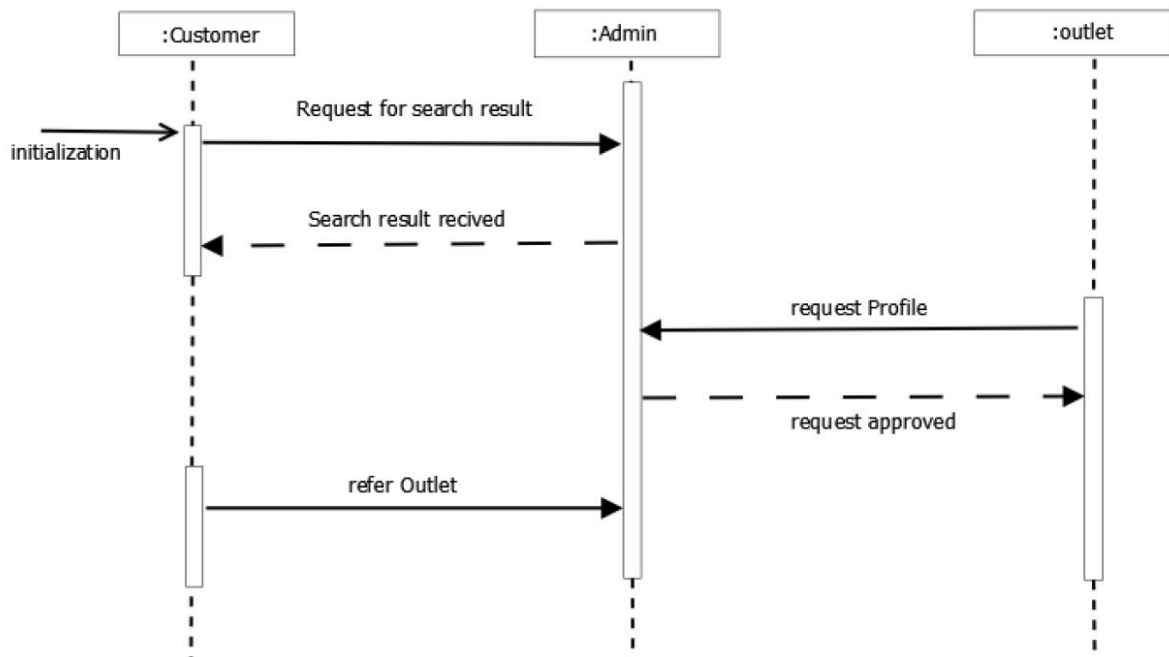
2.4.3 UML SEQUENCE DIAGRAM

A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart. A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios. A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

Sequence diagrams can be useful references for businesses and other organizations. Sequence diagram helps to:

- Represent the details of a UML use case.
- Model the logic of a sophisticated procedure, function, or operation.
- See how objects and components interact with each other to complete a process.
- Plan and understand the detailed functionality of an existing or future scenario.



2.5 SYSTEM SPECIFICATION

2.5.1 HARDWARE SPECIFICATION

Processor - Pentium IV/AMD Dual core

RAM - 1 GB

Hard disk - 500 GB

2.5.2 SOFTWARE SPECIFICATION

Front End - PHP

Backend - MYSQL

Client on PC - Windows 10

Technologies used - JavaScript, HTML5, AJAX, jQuery, PHP, Laravel, CSS, Bootstrap

2.6 SOFTWARE DESCRIPTION

2.6.1 PHP

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. PHP is now installed on more than 244 million websites and 2.1 million web servers. Originally created by Rasmus Lerdorf in 1995, the reference implementation of PHP is now produced by the PHP group. While PHP originally stood for personal Home page, it now stands for PHP: Hypertext Preprocessor, a recursive acronym.

PHP is a server-side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites. It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server. It is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time. It supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.

PHP code is interpreted by a web server with a PHP processor module which generates the resulting web page. Commands can be embedded directly into a HTML source document rather than calling an external file to process data. It has also evolved to include a command-line interface capability and can be used in standalone mode. It is compatible with the GNU General Public License (GPL) due to restrictions on the usage of the term PHP. PHP can be deployed on most web servers and also as a standalone shell on almost every operating system and platform, free of charge.

2.6.2 MySQL

MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation.

MySQL is a database management system.

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

MySQL databases are relational.

A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one-to-one, one-to-many, unique, required or optional, and “pointers” between different tables. The database enforces these rules, so that with a well-designed database, your application never sees inconsistent, duplicate, orphan, out-of-date, or missing data.

The SQL part of “MySQL” stands for “Structured Query Language”. SQL is the most common standardized language used to access databases. Depending on your programming environment, you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language,

or use a language- specific API that hides the SQL syntax. SQL is defined by the ANSI/ISO SQL Standard. The SQL standard has been evolving since 1986 and several

versions exist. In this manual, “SQL92” refers to the standard released in 1992, “SQL:1999” refers to the standard released in 1999, and “SQL:2003” refers to the current version of the standard. We use the phrase “the SQL standard” to mean the current version of the SQL Standard at any time.

MySQL software is Open Source.

Open Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything. If you wish, you may study the source code and change it to suit your needs. The MySQL software uses the GPL (GNU General Public License), to define what you may and may not do with the software in different situations. The MySQL Database Server is very fast, reliable, scalable, and easy to use.

MySQL Server can run comfortably on a desktop or laptop, alongside your other applications, web servers, and so on, requiring little or no attention. If you dedicate an entire machine to MySQL, you can adjust the settings to take advantage of all the memory, CPU power, and I/O capacity available. MySQL can also scale up to clusters of machines, networked together.

MySQL Server was originally developed to handle large databases much faster than existing solutions and has been successfully used in highly demanding production environments for several years. Although under constant development, MySQL Server today offers a rich and useful set of functions. Its connectivity, speed, and security make MySQL Server highly suited for accessing databases on the Internet.

MySQL Server works in client/server or embedded systems.

The MySQL Database Software is a client/server system that consists of a multi-threaded SQL server that supports different backends, several different client programs and libraries, administrative tools, and a wide range of application

programming interfaces (APIs).

We also provide MySQL Server as an embedded multi-threaded library that you can link into your application to get a smaller, faster, easier-to-manage standalone product.

A large amount of contributed MySQL software is available.

MySQL Server has a practical set of features developed in close cooperation with our users. It is very likely that your favorite application or language supports the MySQL Database Server.

2.6.3 LARAVEL

Laravel is a free, open-source PHP web framework, created by Taylor Otwell and intended for the development of web applications following the model–view–controller (MVC) architectural pattern and based on Symfony. Some of the features of Laravel are a modular packaging system with a dedicated dependency manager, different ways for accessing relational databases, utilities that aid in application deployment and maintenance, and its orientation toward syntactic sugar.

Laravel aims to make the development process a pleasing one for the developer without sacrificing application functionality. 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.

Laravel offers you the following advantages, when you are designing a web application based on it –

- The web application becomes more scalable, owing to the Laravel framework.

- Considerable time is saved in designing the web application, since Laravel reuses the components from other frameworks in developing web application.
- It includes namespaces and interfaces, thus helps to organize and manage resources.

Composer

Composer is a tool which includes all the dependencies and libraries. It allows a user to create a project with respect to the mentioned framework (for example, those used in Laravel installation). Third party libraries can be installed easily with help of composer. All the dependencies are noted in composer.json file which is placed in the source folder.

Artisan

Command line interface used in Laravel is called Artisan. It includes a set of commands which assists in building a web application. These commands are incorporated from Symphony framework.

Laravel offers the following key features which makes it an ideal choice for designing web applications –

Modularity

Laravel provides 20 built in libraries and modules which helps in enhancement of the application. Every module is integrated with Composer dependency manager which eases updates.

Testability

Laravel includes features and helpers which helps in testing through various test cases. This feature helps in maintaining the code as per the requirements.

Routing

Laravel provides a flexible approach to the user to define routes in the web application. Routing helps to scale the application in a better way and increases its performance.

Configuration Management

A web application designed in Laravel will be running on different environments, which means that there will be a constant change in its configuration. Laravel provides a consistent approach to handle the configuration in an efficient way.

Query Builder and ORM

Laravel incorporates a query builder which helps in querying databases using various simple chain methods. It provides ORM (Object Relational Mapper) and Active Record implementation called Eloquent.

Schema Builder

Schema Builder maintains the database definitions and schema in PHP code. It also maintains a track of changes with respect to database migrations.

Template Engine

Laravel uses the Blade Template engine, a lightweight template language used to design hierarchical blocks and layouts with predefined blocks that include dynamic content.

E-mail

Laravel includes a mail class which helps in sending mail with rich content and attachments from the web application.

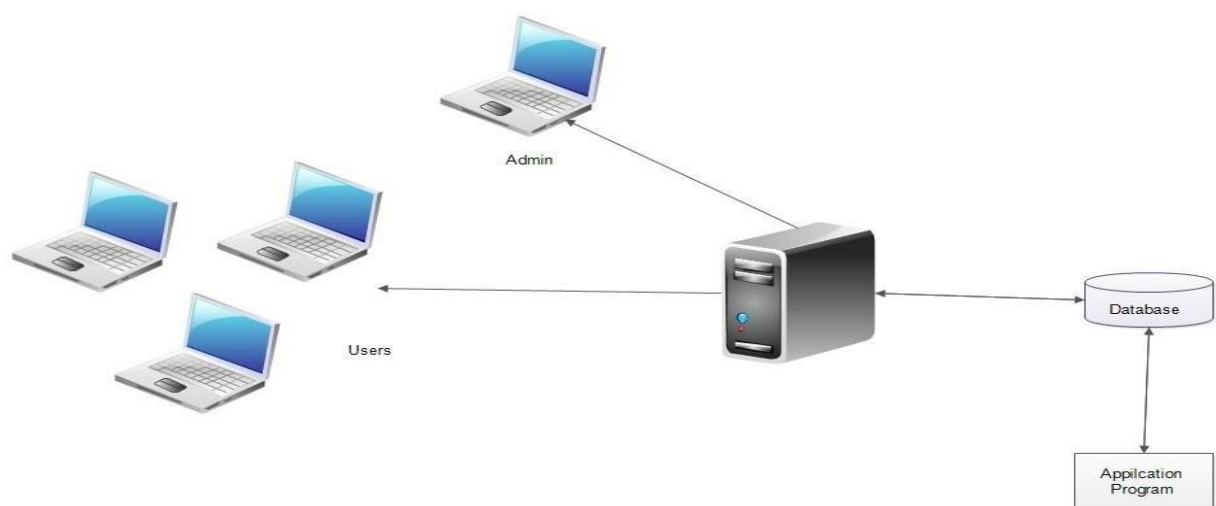
Authentication

User authentication is a common feature in web applications. Laravel eases designing authentication as it includes features such as register, forgot password and send password reminders.

2.7 SYSTEM DESIGN

Design is the first step into the development phase for any engineered product or system. Design is a creative process. A good design is the key to effective system. The term “design” is defined as “the process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization”. It may be defined as a process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization. Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm that is used. The system design develops the architectural detail required to build a system or product. As in the case of any systematic approach, this software too has undergone the best possible design phase fine tuning all efficiency, performance and accuracy levels. The design phase is a transition from a user-oriented document to a document to the programmers or database personnel. System design goes through two phases of development: Logical and Physical Design

2.7.1 ARCHITECTURAL DESIGN



The registered user, admin, Outlet owners and guest users can access Knowhere through internet using their Laptop, Smart Phone, Tablet or Desktop Computer. The

System's application program processes the user's request and provides the required services by taking data from the system database.

2.7.2 MODULE DESIGN

Admin Module

The administrator of the company is allowed to access all the services in the system. User management, Outlet management, Approve listing request from Owners.

- Manager and controller of the entire system.
- Approves Vendors profile
- Complete user management
- Adding categories for vendors listing

Outlet owner module

- Each vendor has a profile
- Initially submit necessary details to the admin and admin verifies the genuinity of the vendor and approves the profile
- Can add /update/edit profile
- Add description and photos
- Interact with user by answering their queries
- Can display special offers, pricing, Availability, and services as ads

Registered User Module

- Can create an account in the application for requesting online assistance from the vendor. Even though users don't need to have an account for browsing or to review and rate vendors.
- Users without an account is validated by some verification for gaining access to the system (either by google login or email/Phone verification)

- Review and rate vendors
- Communicate with vendors
- Point out or refer missing vendors

Guest User Module

- No need for registering with Knowhere
- Can use search function
- Post comments and reviews and also rate

2.7.3 UML DIAGRAMS

UML is an acronym that stands for Unified Modeling Language. Simply put, UML is a modern approach to modeling and documenting software. In fact, it's one of the most popular business process modeling techniques.

It is based on diagrammatic representations of software components. By using visual representations, we are able to better understand possible flaws or errors in software or business processes.

UML was created as a result of the chaos revolving around software development and documentation. In the 1990s, there were several different ways to represent and document software systems. The need arose for a more unified way to visually represent those systems and as a result, in 1994-1996, the UML was developed by three software engineers working at Rational Software. It was later adopted as the standard in 1997 and has remained the standard ever since.

Mainly, UML has been used as a general-purpose modeling language in the field of software engineering. However, it has now found its way into the documentation of several business processes or workflows.

There are several types of UML diagrams and each one of them serves a different purpose regardless of whether it is being designed before the implementation or after (as part of documentation).

The two most broad categories that encompass all other types are Behavioral UML

diagram and Structural UML diagram. As the name suggests, some UML diagrams try to analyze and depict the structure of a system or process, whereas others describe the behavior of the system, its actors, and its building components. The different types are broken down as follows:

Structural Diagrams

The structural diagrams represent the static aspect of the system. These static aspects represent those parts of a diagram, which forms the main structure and are therefore stable. The structural diagrams are –

- Class diagram
- Object diagram
- Component diagram

Behavioral Diagrams

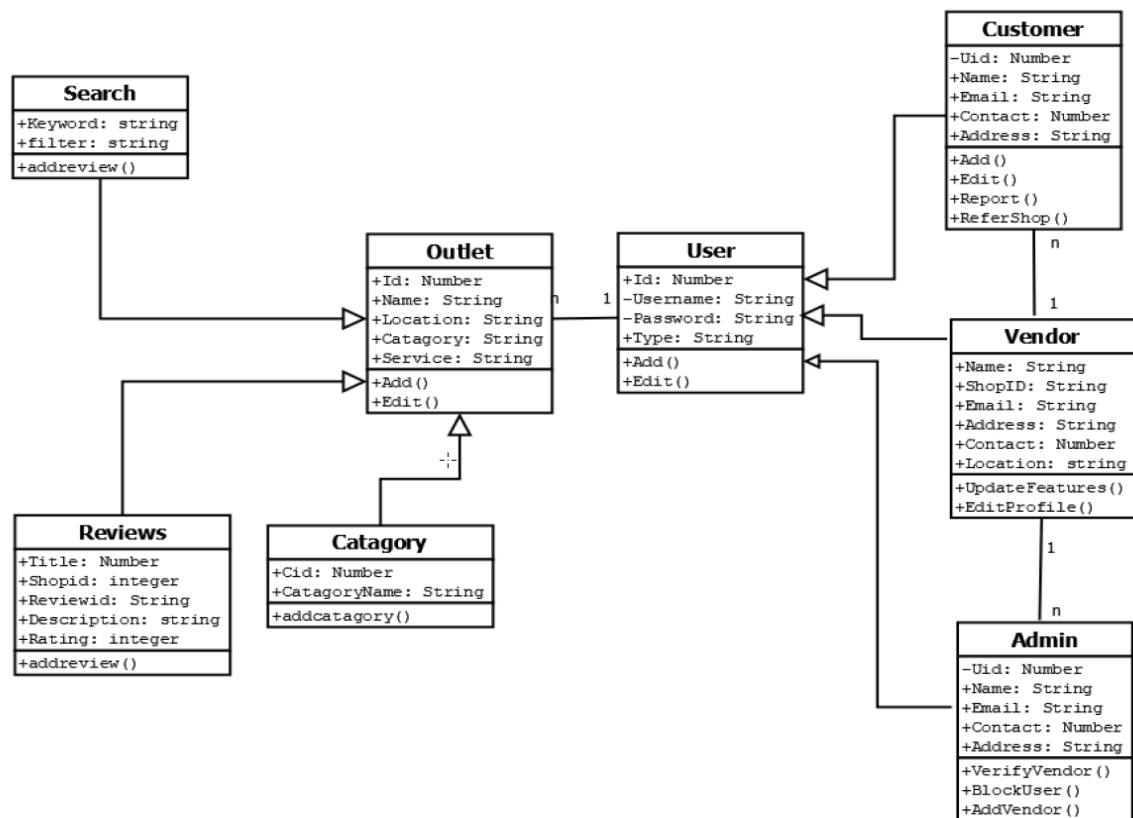
Behavioral diagrams basically capture the dynamic aspect of a system. Dynamic aspect can be further described as the changing/moving parts of a system. UML has the following types of behavioral diagrams –

- Use case diagram
- Sequence diagram
- State chart diagram
- Activity diagram

2.7.3.1 CLASS DIAGRAM

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interfaces, associations, and collaboration. Class diagrams basically represent the object-oriented view of a system, which is static in nature.

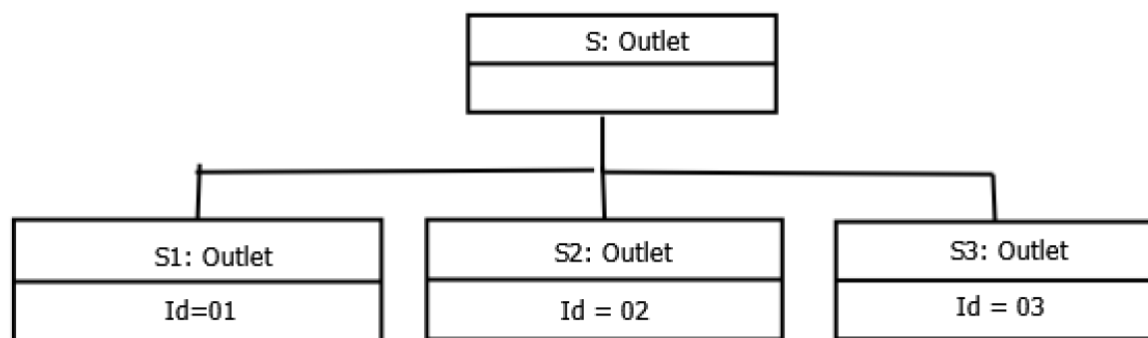
Class diagram represents the object orientation of a system. Hence, it is generally used for development purpose. This is the most widely used diagram at the time of system construction.



2.7.3.2 OBJECT DIAGRAM

Object diagrams can be described as an instance of class diagram. Thus, these diagrams are closer to real-life scenarios where we implement a system. Object diagrams are a set of objects and their relationship is just like class diagrams. They also represent the static view of the system.

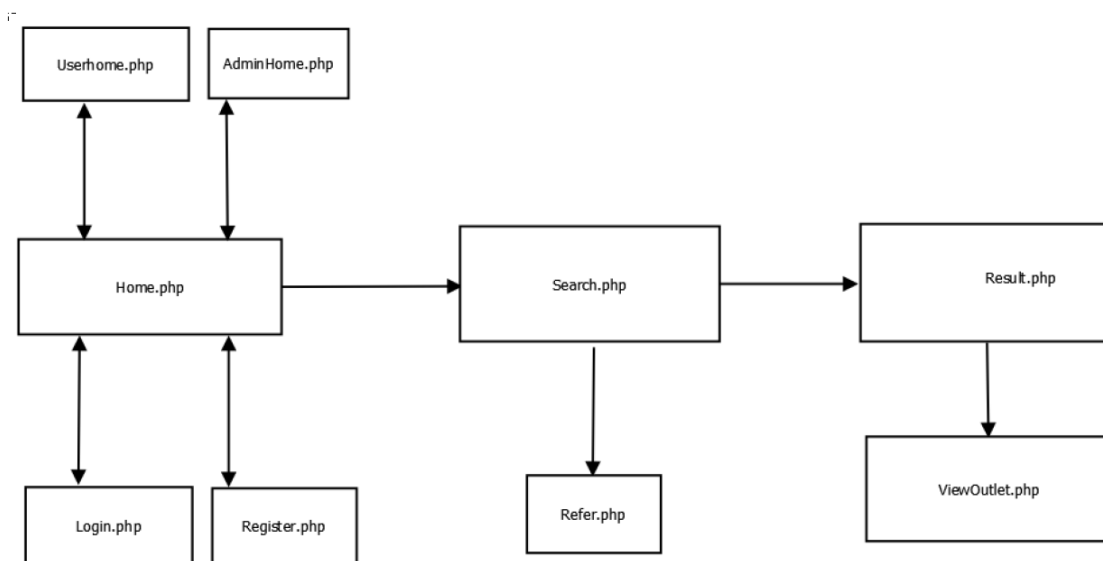
The usage of object diagrams is similar to class diagrams but they are used to build prototype of a system from a practical perspective.



2.7.3.3 COMPONENT DIAGRAM

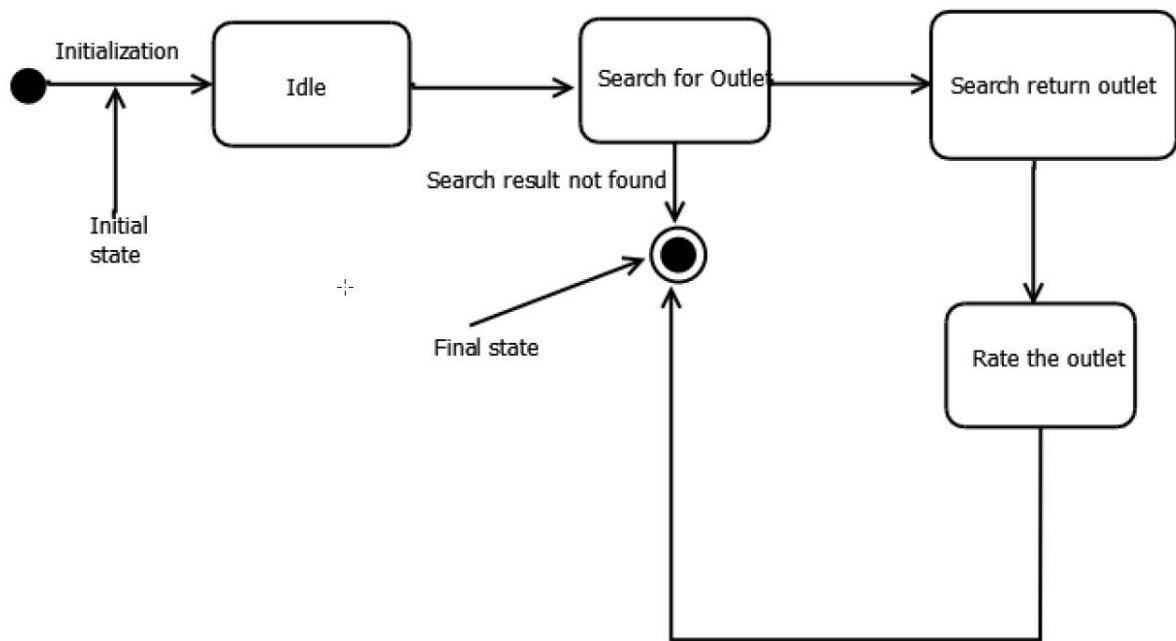
Component diagrams represent a set of components and their relationships. These components consist of classes, interfaces, or collaborations. Component diagrams represent the implementation view of a system.

During the design phase, software artifacts (classes, interfaces, etc.) of a system are arranged in different groups depending upon their relationship. Now, these groups are known as components.



2.7.3.4 STATE CHART DIAGRAM

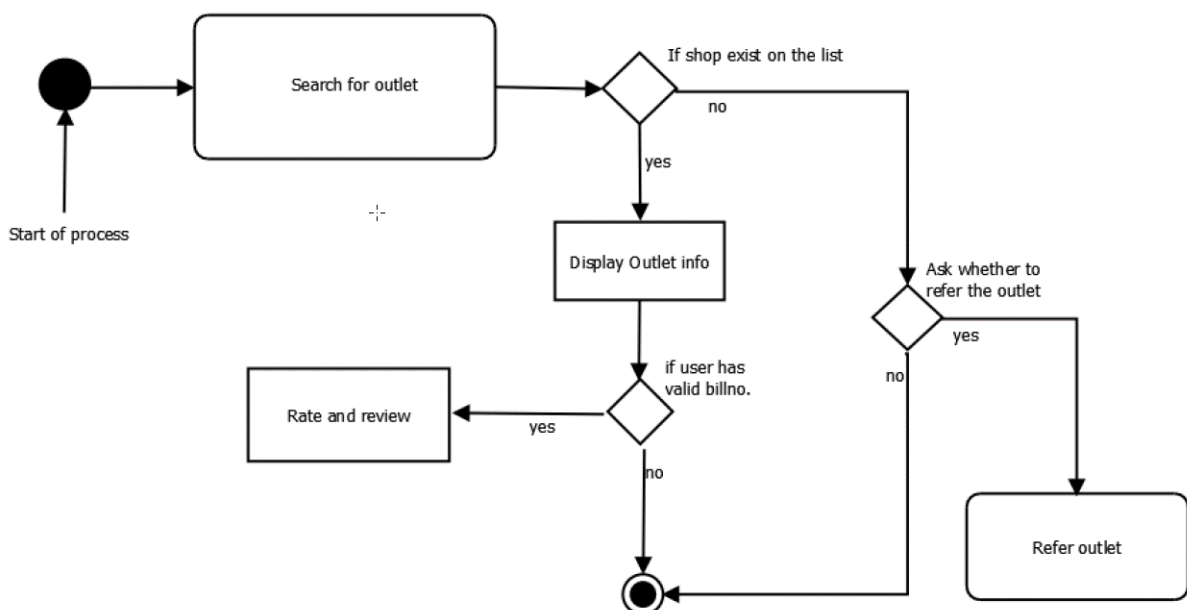
Any real-time system is expected to be reacted by some kind of internal/external events. These events are responsible for state change of the system. State chart diagram is used to represent the event driven state change of a system. State chart diagram is used to visualize the reaction of a system by internal/external factors.



2.7.3.5 ACTIVITY DIAGRAM

Activity diagram describes the flow of control in a system. It consists of activities and links. The flow can be sequential, concurrent, or branched. Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system.

Activity diagrams are used to visualize the flow of controls in a system. This is prepared to have an idea of how the system will work when executed.



2.7.4 DATABASE DESIGN

A database is an organized mechanism that has the capability of storing information through which a user can retrieve stored information in an effective and efficient manner. The data is the purpose of any database and must be protected.

The database design is a two-level process. In the first step, user requirements are gathered together and a database is designed which will meet these requirements as clearly as possible. This step is called Information Level Design and it is taken independent of any individual DBMS.

In the second step, this Information level design is transferred into a design for the specific DBMS that will be used to implement the system in question. This step is called Physical Level Design, concerned with the characteristics of the specific DBMS that will be used. A database design runs parallel with the system design. The organization of the data in the database is aimed to achieve the following two major objectives.

Data Integrity:

Data integrity refers to the accuracy, consistency, and reliability of data that is stored in the database. Both database designers and database developers are responsible for implementing data integrity within one or a set of related databases.

- Domain integrity rules
- Entity integrity rules
- Referential integrity rules

Data independence:

Data independence is the type of data transparency that matters for a centralized DBMS. It refers to the immunity of user applications to changes made in the definition and organization of data. Application programs should not, ideally, be exposed to details of data representation and storage. The DBMS provides an abstract view of the data that hides such details.

- Physical data independence
- Logical data independence

Relational Database Management System (RDBMS)

A relational model represents the database as a collection of relations. Each relation resembles a table of values or file of records. In formal relational model terminology, a row is called a tuple, a column header is called an attribute and the table is called a relation. A relational database consists of a collection of tables, each of which is assigned a unique name. A row in a table represents a set of related values.

Relations, Domains & Attributes

A table is a relation. The rows in a table are called tuples. A tuple is an ordered set of n elements. Columns are referred to as attributes. Relationships have been set between every table in the database. This ensures both Referential and Entity Relationship Integrity. A domain D is a set of atomic values. A common method of specifying a domain is to specify a data type from which the data values forming the domain are drawn. It is also useful to specify a name for the domain to help in interpreting its values. Every value in a relation is atomic, that is not decomposable.

Relationships

- Table relationships are established using Key. The two main keys of prime importance are Primary Key & Foreign Key. Entity Integrity and Referential

Integrity Relationships can be established with these keys.

- Entity Integrity enforces that no Primary Key can have null values.
- Referential Integrity enforces that no Primary Key can have null values.
- Referential Integrity for each distinct Foreign Key value, there must exist a matching Primary Key value in the same domain. Other key is Super Key and Candidate Keys.

Normalization

Data are grouped together in the simplest way so that later changes can be made with minimum impact on data structures. Normalization is formal process of data structures in manners that eliminates redundancy and promotes integrity. Normalization is a technique of separating redundant fields and breaking up a large table into a smaller one. It is also used to avoid insertion, deletion, and updating anomalies. Normal form in data modelling use two concepts, keys and relationships. A key uniquely identifies a row in a table. There are two types of keys, primary key and foreign key. A primary key is an element or a combination of elements in a table whose purpose is to identify records from the same table. A foreign key is a column in a table that uniquely identifies record from a different table. All the tables have been normalized up to the third normal form.

As the name implies, it denotes putting things in the normal form. The application developer via normalization tries to achieve a sensible organization of data into proper tables and where names can be easily correlated to the data by the user. Normalization eliminates repeating groups and thereby avoids data redundancy which proves to be a great burden on the computer resources. These include:

- Normalize the data.
- Choose proper names for the tables and columns.
- Choose the proper name for the data.

First Normal Form

The First Normal Form states that the domain of an attribute must include only atomic values and that the value of any attribute in a tuple must be a single value from the domain of that attribute. In other words, 1NF disallows “relations within relations” or “relations as attribute values within tuples”. The only attribute values permitted by 1NF are single atomic or indivisible values. The first step is to put the data into First Normal Form. This can be done by moving data into separate tables where the data is of similar type in each table. Each table is given a Primary Key or Foreign Key as per requirement of the project. In this we form new relations for each non-atomic attribute or nested relation. This eliminated repeating groups of data. A relation is said to be in first normal form if only if it satisfies the constraints that contain the primary key only.

Second Normal Form

According to Second Normal Form, for relations where primary key contains multiple attributes, no non-key attribute should be functionally dependent on a part of the primary key. In this we decompose and setup a new relation for each partial key with its dependent attributes. Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it. This step helps in taking out data that is only dependent on a part of the key. A relation is said to be in second normal form if and only if it satisfies all the first normal form conditions for the primary key and every non-primary key attributes of the relation is fully dependent on its primary key alone.

Third Normal Form

According to Third Normal Form, Relation should not have a non-key attribute functionally determined by another non-key attribute or by a set of non-key attributes. That is, there should be no transitive dependency on the primary key. In

this we decompose and set up relation that includes the non-key attributes that functionally determines other non-key attributes. This step is taken to get rid of anything that does not depend entirely on the Primary Key. A relation is said to be in third normal form if only if it is in second normal form and more over the non key attributes of the relation should not be depend on another non-key attribute.

2.7.4.1 Table design in database

- Table structure for table password_resets

Column	Type	Null	Key
<i>id</i>	int(11)	No	Primary key
email	varchar(191)	No	
token	varchar(191)	No	
created_at	timestamp	Yes	

- Table structure for table tbl_advert

Column	Type	Null	Key
<i>ad_id</i>	int(10)	No	Primary key
id	int(10)	No	Foreign key from tbl_login
outletid	bigint(20)	No	Foreign key from tbl_outlet_prof

ad_content	varchar(191)	Yes	
description	varchar(500)	Yes	
pkg_id	int(10)	No	Foreign key from tbl_packages
validity	int(20)	No	
expiring_in	int(20)	Yes	
status_id	int(10)	No	Foreign key from tbl_status
p_status	int(10)	No	Foreign key from tbl_payments

- Table structure for table tbl_cat

Column	Type	Null	Key
<i>Cat_id</i>	int(10)	No	Primary key

catagory	varchar(191)	No
created_at	timestamp	Yes
updated_at	timestamp	Yes

- Table structure for table tbl_city

Column	Type	Null	Key
<i>city_id</i>	int(10)	No	Primary key
dist_id	int(10)	No	Foreign key from tbl_district
city	varchar(191)	No	
created_at	timestamp	Yes	
updated_at	timestamp	Yes	

- Table structure for table tbl_district

Column	Type	Null	Key
<i>dist_id</i>	int(10)	No	Primary key
state_id	int(10)	No	Foreign key from tbl_state
district	varchar(191)	No	

created_at	timestamp	Yes
updated_at	timestamp	Yes

- Table structure for table `tbl_listing_rqst`

Column	Type	Null	Key
<i>rqst_id</i>	bigint(20)	No	Primary key
email	varchar(191)	No	
outletname	varchar(191)	No	
city_id	int(10)	No	Foreign key from
phone	varchar(191)	No	
ownername	varchar(191)	No	
subcat_id	int(10)	No	Foreign key from tbl_subcat
Proof1	varchar(191)	No	
Proof2	varchar(191)	No	
Proof3	varchar(191)	No	
status_id	int(10)	No	Foreign key from tbl_status
created_at	timestamp	Yes	

updated_at	timestamp	Yes
-------------------	-----------	-----

- Table structure for table tbl_locality

Column	Type	Null	Key
<i>loc_id</i>	int(10)	No	Primary key
city_id	int(10)	No	Foreign key from tbl_city
locality	varchar(191)	No	
created_at	timestamp	Yes	
updated_at	timestamp	Yes	

- Table structure for table tbl_login

Column	Type	Null	Key
<i>id</i>	int(10)	No	Primary key
email	varchar(191)	No	
password	varchar(191)	No	
utype_id	int(10)	No	Foreign key from tbl_utype
status_id	int(10)	No	Foreign key from tbl_status

verified	tinyint(1)	No
remember_token	varchar(100)	Yes
created_at	timestamp	Yes
updated_at	timestamp	Yes

- Table structure for table tbl_outlet_prof

Column	Type	Null	Key
<i>outletid</i>	bigint(20)	No	Primary key
id	int(10)	No	Foreign key from tbl_login
regid	int(10)	No	Foreign key from tbl_users_reg
outletname	varchar(191)	No	
ownername	varchar(191)	No	
address	varchar(191)	Yes	
latitude	double	Yes	
longitude	double	Yes	
otitle	text	Yes	
description	longtext	Yes	

website	varchar(150)	Yes	
oemail	varchar(191)	Yes	
city_id	int(10)	No	Foreign key from tbl_city
subcat_id	int(10)	No	Foreign key from tbl_subcat
Service_id	varchar(50)	Yes	Foreign key from tbl_services
phone1	varchar(191)	No	
phone2	varchar(191)	Yes	
rcount	int(10)	No	
status_id	int(10)	No	Foreign key from tbl_status
created_at	timestamp	Yes	
updated_at	timestamp	Yes	

- Table structure for table tbl_package

Column	Type	Null	Key
<i>pkg_id</i>	int(10)	No	Primary key
pkg_name	varchar(30)	No	
duration	int(11)	No	

amount	int(11)	No
---------------	---------	----

- Table structure for table tbl_payments

Column	Type	Null	Key
<i>pay_id</i>	int(10)	No	Primary key
id	int(10)	No	Foreign key from
ad_id	int(10)	No	Foreign key from
amount	varchar(191)	No	
cname	varchar(191)	No	
cc_number	varchar(191)	No	
ctype	varchar(191)	Yes	
cvv	varchar(191)	No	
month	varchar(191)	No	
year	varchar(191)	No	

status_id	int(10)	No	Foreign key from
------------------	---------	----	---------------------

- Table structure for table `tbl_prof_images`

Column	Type	Null	Key
<i>imgid</i>	int(11)	No	Primary key
outletid	bigint(20)	No	Foreign key from
imgname	varchar(200)	No	
created_at	timestamp	Yes	
updated_at	timestamp	Yes	

- Table structure for table `tbl_report`

Column	Type	Null	Key
<i>rid</i>	int(10)	No	Primary key
outletid	bigint(20)	No	Foreign key from tbl_outlet_prof
email	varchar(191)	No	

name	varchar(191)	No
reason	varchar(191)	No
created_at	timestamp	Yes
updated_at	timestamp	Yes

- Table structure for table tbl_review

Column	Type	Null	Key
<i>rev_id</i>	int(10)	No	Primary key
email	varchar(191)	No	
name	varchar(191)	No	
outlet_id	bigint(20)	No	Foreign key from
review	varchar(191)	No	
rating	double(8,2)	No	

title	varchar(191)	No
created_at	timestamp	Yes
updated_at	timestamp	Yes

- Table structure for table `tbl_services`

Column	Type	Null	Key
<i>service_id</i>	int(10)	No	Primary key
service	varchar(191)	No	
Subcat_id	int(10)	No	Foreign key from <code>tbl_subcatct</code>
created_at	timestamp	Yes	
updated_at	timestamp	Yes	

- Table structure for table `tbl_state`

Column	Type	Null	Key
<i>state_id</i>	int(10)	No	Primary key

state	varchar(191)	No
created_at	timestamp	Yes
updated_at	timestamp	Yes

- Table structure for table tbl_status

Column	Type	Null	Key
<i>status_id</i>	int(10)	No	Primary key
status	varchar(191)	No	
created_at	timestamp	Yes	
updated_at	timestamp	Yes	

- Table structure for table tbl_subcat

Column	Type	Null	Key
<i>subcat_id</i>	int(10)	No	Primary key

cat_id	int(10)	No	Foreign key from tbl_cat
subcatagory	varchar(191)	No	
created_at	timestamp	Yes	
updated_at	timestamp	Yes	

- Table structure for table tbl_suggest

Column	Type	Null	Key
<i>sid</i>	int(10)	No	Primary key
outletid	bigint(20)	No	Foreign key from
uid	int(10)	No	Foreign key from tbl_login
email	varchar(191)	No	
comment	varchar(191)	No	
response	varchar(191)	Yes	
subject	varchar(191)	No	
rstatus	int(10)	No	Foreign key from tbl_status

created_at	timestamp	Yes
updated_at	timestamp	Yes

- Table structure for table `tbl_users_reg`

Column	Type	Null	Key
<i>regid</i>	int(10)	No	Primary key
id	int(10)	No	Foreign key from
name	text	No	
city_id	int(10)	Yes	Foreign key from <code>tbl_city</code>
phone	varchar(191)	No	
title	text	Yes	
oaddress	text	Yes	
image	varchar(191)	Yes	
status_id	int(10)	No	Foreign key from <code>tbl_status</code>
created_at	timestamp	Yes	
updated_at	timestamp	Yes	

- Table structure for table `tbl_ctype`

Column	Type	Null	Key
<i>ctype_id</i>	int(10)	No	Primary key
ctype	varchar(191)	No	
created_at	timestamp	Yes	
updated_at	timestamp	Yes	

- Table structure for table `tbl_verify_mail`

Column	Type	Null	Key
<i>eid</i>	int(10)	No	Primary key
email	varchar(191)	No	
code	varchar(191)	No	
isverified	int(10)	No	Foreign key from <code>tbl_status</code>

created_at	timestamp	Yes
updated_at	timestamp	Yes

- Table structure for table verify_users

Column	Type		Key
<i>vid</i>	bigint(20)	No	Primary key
id	int(10)	No	Foreign key from
token	varchar(191)	No	
created_at	timestamp	Yes	
updated_at	timestamp	Yes	

2.8 SYSTEM TESTING

2.8.1 INTRODUCTION

Software Testing is the process of executing software in a controlled manner, in order to answer the question - Does the software behave as specified? Software testing is often used in association with the terms verification and validation. Validation is the checking or testing of items, includes software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques such as reviews, analysis, inspections, and walkthroughs. Validation is the process of checking that what has been specified is what the user actually wanted.

Other activities which are often associated with software testing are static analysis and dynamic analysis. Static analysis investigates the source code of software, looking for problems and gathering metrics without actually executing the code. Dynamic analysis looks at the behavior of software while it is executing, to provide information such as execution traces, timing profiles, and test coverage information.

Testing is a set of activity that can be planned in advanced and conducted systematically. Testing begins at the module level and work towards the integration of entire computers-based system. Nothing is complete without testing, as it vital success of the system testing objectives, there are several rules that can serve as testing objectives. They are:

- Testing is a process of executing a program with the intent of finding an error.
- good test case is one that has high possibility of finding an undiscovered error.
- successful test is one that uncovers an undiscovered error.

If a testing is conducted successfully according to the objectives as stated above, it would uncover errors in the software. Also testing demonstrate that the software function appears to be working according to the specification, that performance requirement appears to have been met.

There are three ways to test program.

- For correctness
- For implementation efficiency
- For computational complexity

Test for correctness are supposed to verify that a program does exactly what it was designed to do. This is much more difficult than it may at first appear, especially for large programs.

2.8.2 TEST PLAN

A test plan implies a series of desired course of action to be followed in accomplishing various testing methods. The Test Plan acts as a blue print for the action that is to be followed. The software engineers create a computer program, its documentation and related data structures. The software developers are always responsible for testing the individual units of the programs, ensuring that each performs the function for which it was designed. There is an independent test group (ITG) which is to remove the inherent problems associated with letting the builder to test the thing that has been built. The specific objectives of testing should be stated in measurable terms. So that the mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours per regression test all should be stated within the test plan.

The levels of testing include:

- Unit testing
- Integration Testing
- Data validation Testing
- Output Testing

2.8.2.1 UNIT TESTING

Unit testing focuses verification effort on the smallest unit of software design – the software component or module. Using the component level design description as a guide, important control paths are tested to uncover errors within the boundary of the module.

The relative complexity of tests and uncovered scope established for unit testing. The unit testing is white-box oriented, and step can be conducted in parallel for multiple components. The modular interface is tested to ensure that information

properly flows into and out of the program unit under test. The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm's execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested.

Tests of data flow across a module interface are required before any other test is initiated. If data do not enter and exit properly, all other tests are moot. Selective testing of execution paths is an essential task during the unit test. Good design dictates that error conditions be anticipated and error handling paths set up to reroute or cleanly terminate processing when an error does occur. Boundary testing is the last task of unit testing step. Software often fails at its boundaries.

Unit testing was done by treating each module as separate entity and testing each one of them with a wide spectrum of test inputs. Some flaws in the internal logic of the modules were found and were rectified. After coding each module is tested and run individually. All unnecessary code were removed and ensured that all modules are working, and gives the expected result.

2.8.2.2 INTEGRATION TESTING

Integration testing is systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design. The entire program is tested as whole. Correction is difficult because isolation of causes is complicated by vast expanse of entire program. Once these errors are corrected, new ones appear and the process continues in a seemingly endless loop.

After performing unit testing in the System all the modules were integrated to test for any inconsistencies in the interfaces. Moreover, differences in program structures were removed and a unique program structure was evolved.

2.8.2.3 VALIDATION TESTING

This is the final step in testing. In this the entire system was tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as Black Box testing or System tests.

Black Box testing method focuses on the functional requirements of the software. That is, Black Box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.

Black Box testing attempts to find errors in the following categories; incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors and initialization errors and termination errors.

2.8.2.4 USER ACCEPTANCE TESTING

The system considered is tested for user acceptance; here it should satisfy the firm's need. The software should keep in touch with perspective system; user at the time of developing and making changes whenever required. This done with respect to the following points:

- Input Screen Designs
- Output Screen Designs

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

2.9 IMPLEMENTATION

Implementation is the stage of the project where the theoretical design is turned into a working system. It can be considered to be the most crucial stage in achieving a successful new system gaining the users confidence that the new system will work and will be effective and accurate. It is primarily concerned with user training and documentation. Conversion usually takes place about the same time the user is being trained or later. Implementation simply means convening a new system design into operation, which is the process of converting a new revised system design into an operational one.

At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned or controlled, it can create chaos and confusion.

Implementation includes all those activities that take place to convert from the existing system to the new system. The new system may be a totally new, replacing an existing manual or automated system or it may be a modification to an existing system. Proper implementation is essential to provide a reliable system to meet organization requirements. The process of putting the developed

system in actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system can be implemented only after through testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system. The more complex the system being implemented, the more involved will be the system analysis and design effort required to implement the three main aspects: education and training, system testing and changeover.

The implementation state involves the following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of methods to achieve the changeover. Training of the staff in the changeover phase.

2.9.1 IMPLEMENTATION PROCEDURE

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended uses and the operation of the system. In many organizations someone who will not be operating it, will commission the software development project. In the initial stage people doubt about the software but we have to ensure that the resistance does not build up, as one has to make sure that:

- The active user must be aware of the benefits of using the new system. □ Their confidence in the software is built up.
- Proper guidance is imparted to the user so that he is comfortable in using the application.

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not up running on the server, the actual process won't take place.

2.9.2 USER TRAINING

User training is designed to prepare the user for testing and converting the system. To achieve the objective and benefits expected from computer-based system, it is essential for the people who will be involved to be confident of their role in the new system. As system becomes more complex, the need for training is more important. By user training the user comes to know how to enter data, respond to error messages, interrogate the database and call up routine that will produce reports and perform other necessary functions.

After providing the necessary basic training on computer awareness the user will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the date entered. It should then cover information needed by the specific user/ group to use the system or part of the system while imparting the training of the program on the application. This training may be different across different user groups and across different levels of hierarchy.

2.9.3 OPERATIONAL DOCUMENT

After providing the necessary basic training on computer awareness the user will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the date entered. It should then cover information needed by the specific user/ group to use the system or part of the system while imparting the training of the program on the application. This training may be different across different user groups and across different levels of hierarchy.

2.9.4 SYSTEM MAINTENANCE

Maintenance is the enigma of system development. The maintenance phase of the software cycle is the time in which a software product performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is for it to make adaptable to the changes in the system environment. Software maintenance is of course, far more than "Finding Mistakes".

2.10 CONCLUSION & FUTURE ENHANCEMENTS

- The system is designed in such a way that the payment of service provider should be done in completely online mode.
- Provide more security.
- Provide Users with the functionality to know stock status, que status and more from vendors.

2.10.1 CONCLUSION

The software reduces the time consumption and the manual efforts of searching a product. It will be a simple platform for users to access services for their huge needs.

The benefits, we can obtain from the new system are:

- Timely and accurate information will be available
- Reduced data loss
- The access time and process time is highly reduced
- Quick data view
- Error free output

The proposed system is expected to replace manual system and provide more efficient performance and services.

2.11 BIBLIOGRAPHY BOOKS/REFERENCES:

- Gary B. Shelly, Harry J. Rosenblatt, *“System Analysis and Design”*, 2009.
- Roger S Pressman, *“Software Engineering”*, 1994.
- PankajJalote, *“Software engineering: a precise approach”*, 2006.
- James lee and Brent ware Addison, *“Open source web development with LAMP”*, 2003
- IEEE Std 1016 Recommended Practice for Software Design Descriptions.

WEBSITES:

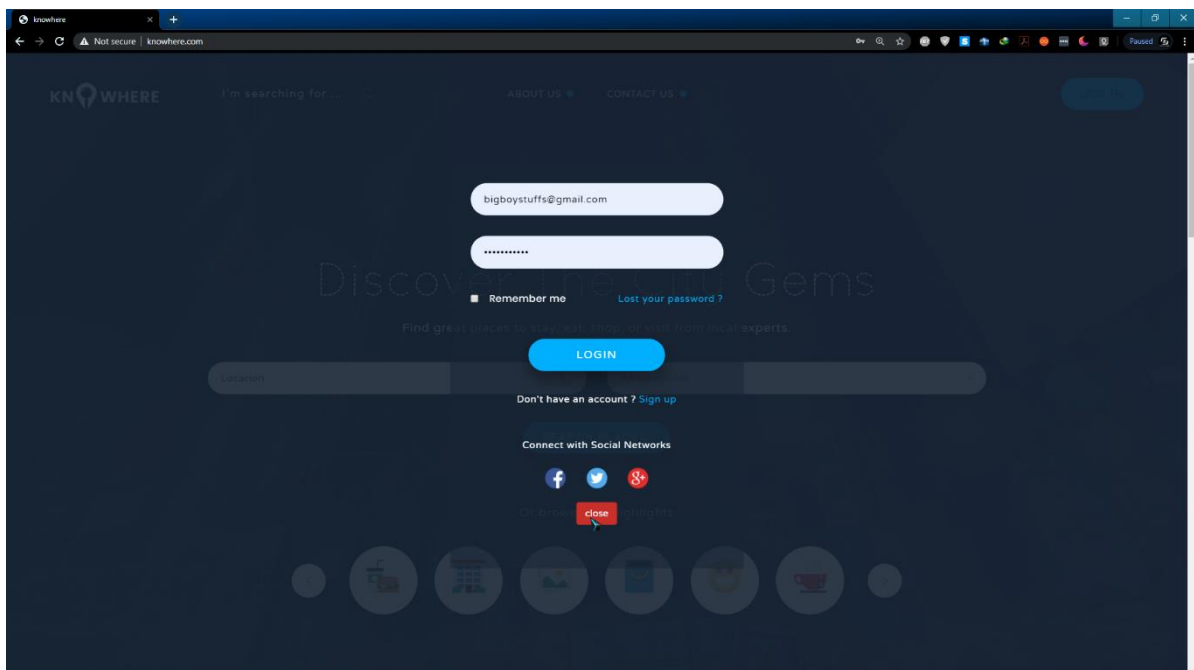
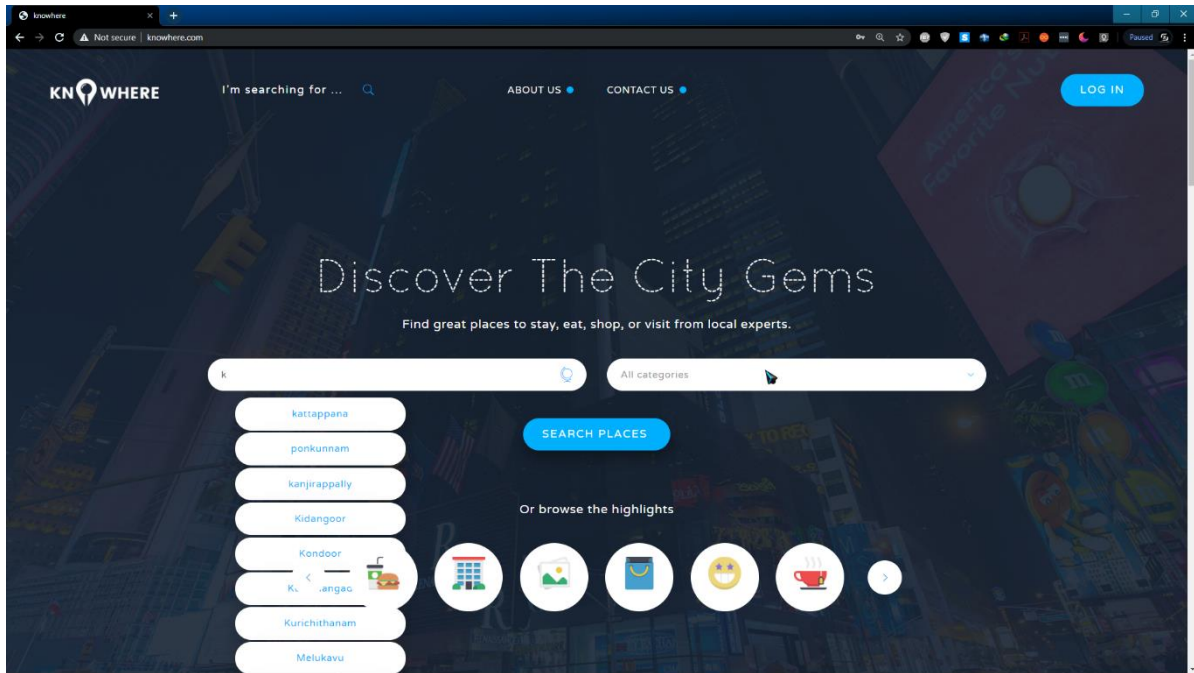
<https://www.tutorialspoint.com/uml/>

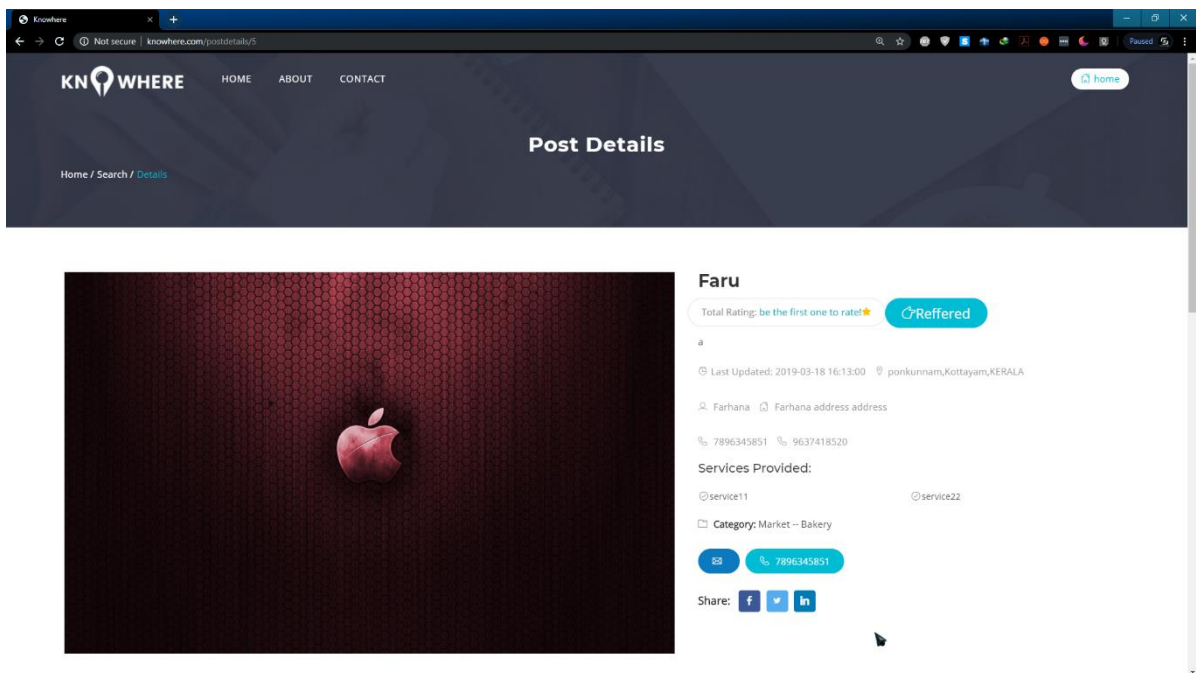
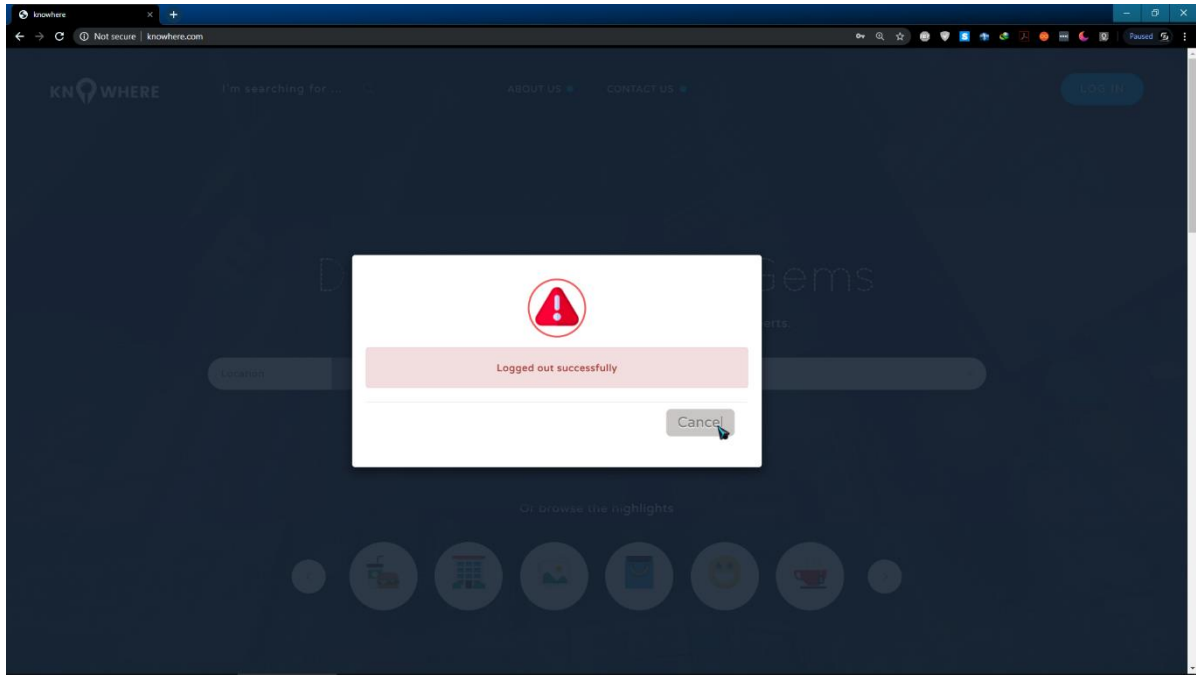
https://www.w3.org/wiki/Main_Page

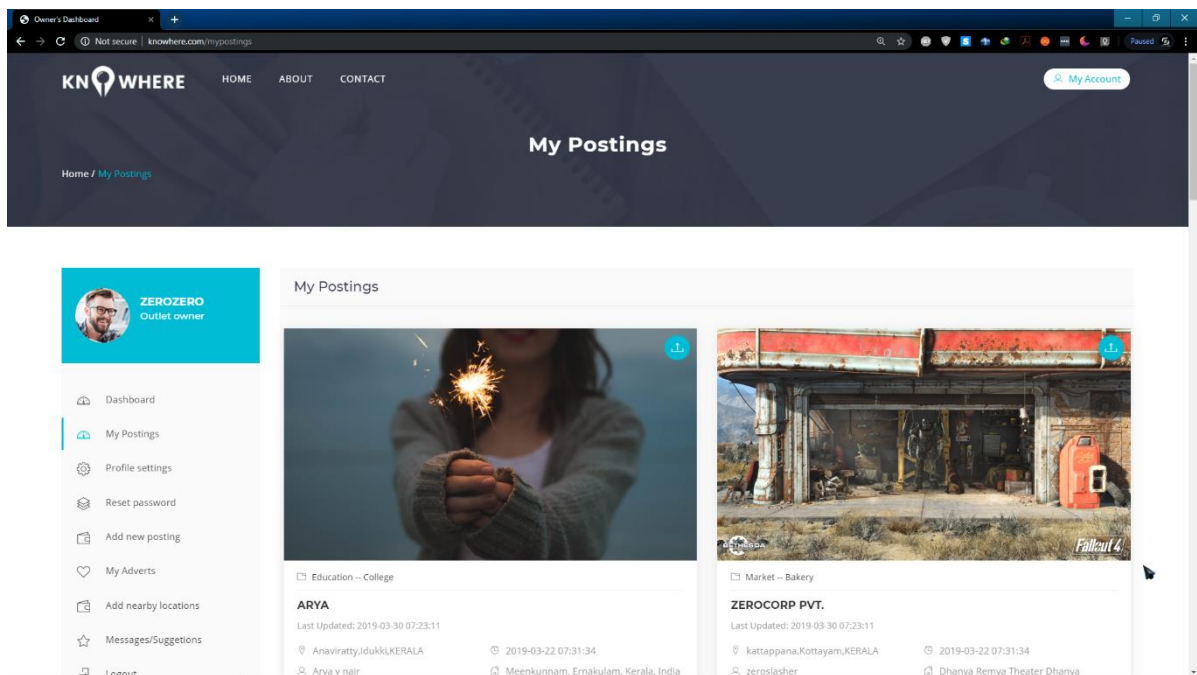
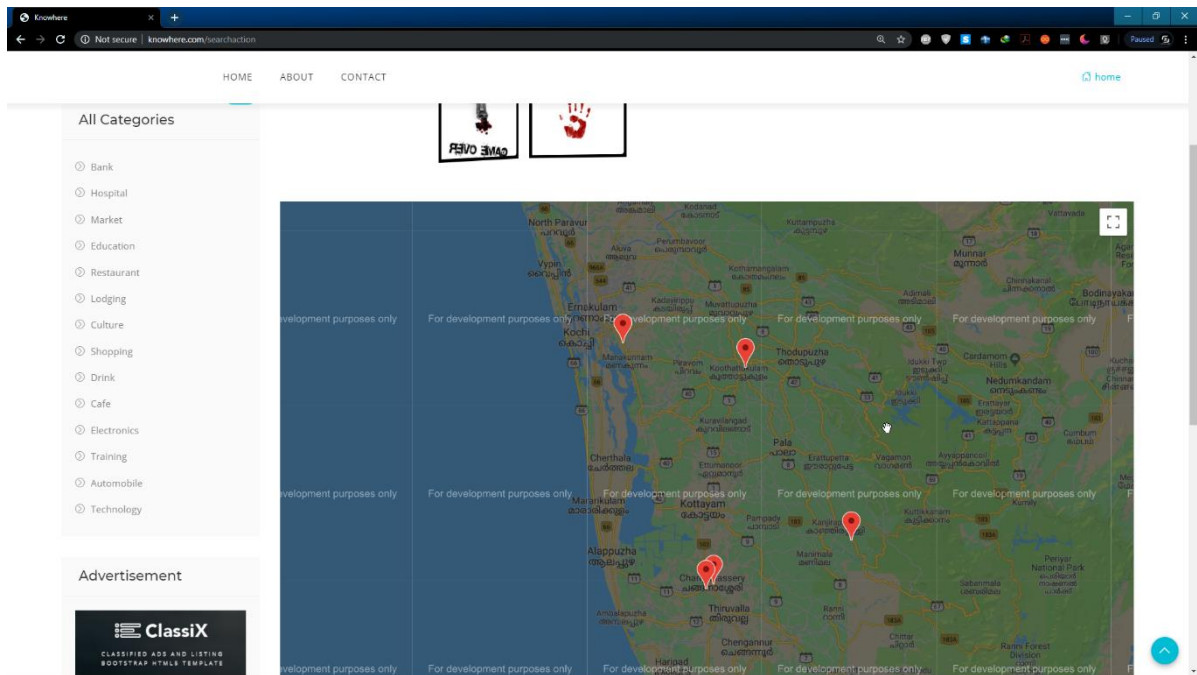
<https://www.javatpoint.com/dbms-normalization>

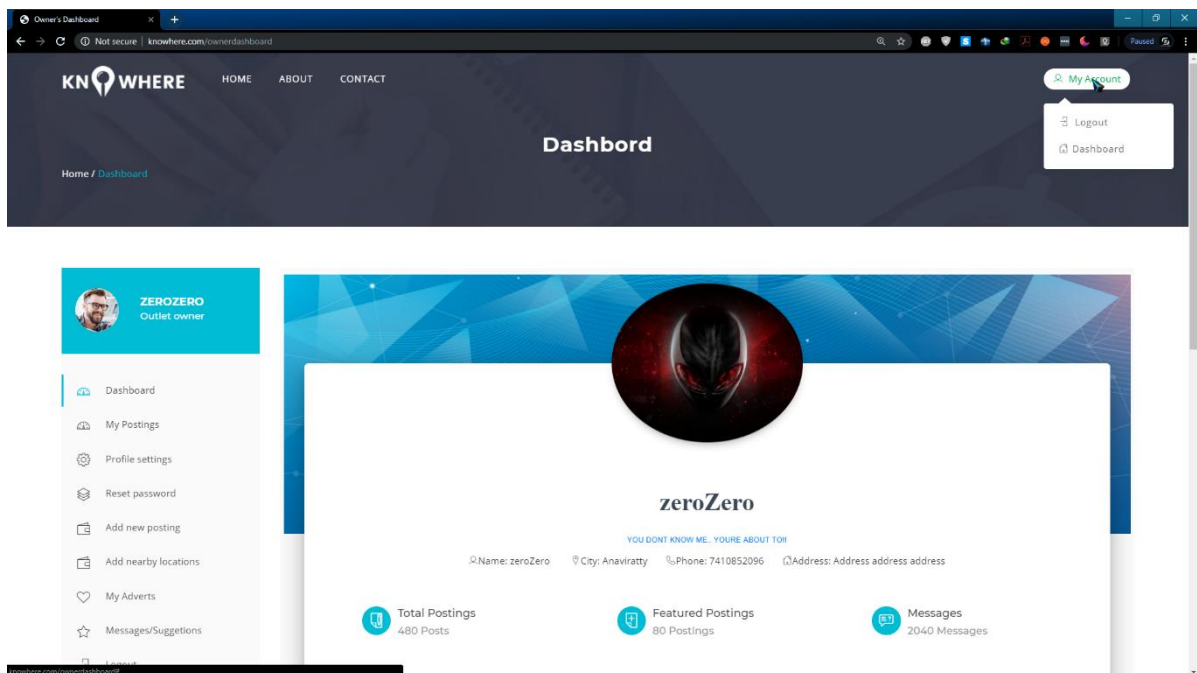
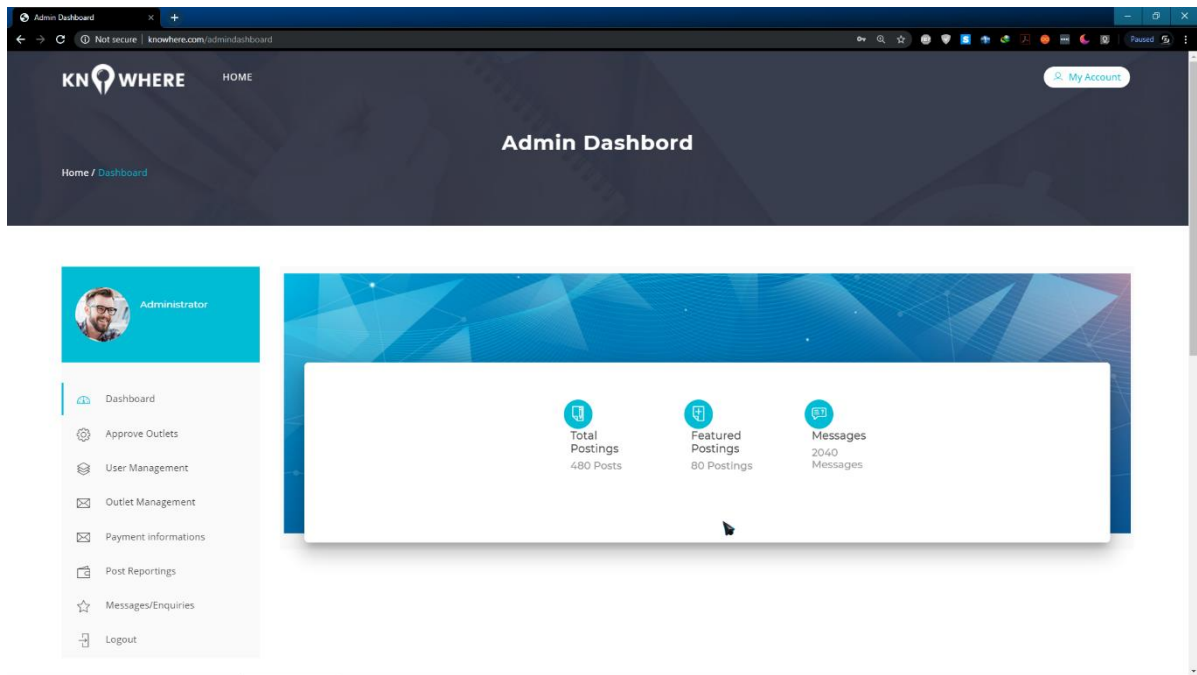
2.11 APPENDIX

2.10.2 SCREENSHOTS









KNOWHERE HOME [My Account](#)

Profile Settings

Home / Profile Settings

USER
Administrator

- Dashboard
- Approve Outlets
- User Management
- Outlet Management
- Post Reportings
- Payment informations
- Messages/Enquiries
- Logout

Dashboard

Total Outlet Owners
5 Owners

Blocked Owners
0 Owners

Active owners
5 owners

Owner name	email	phone	Location	Address	Status	Actions
zeroZero	bigboystuffs@gmail.com	7410852096	Anaviratty District -- State	Address address address	ACTIVE	
John	usernotalive@gmail.com	8234567890	kattappana District -- State	middle of Nowhere	ACTIVE	

KNOWHERE HOME ABOUT CONTACT [My Account](#)

Add new posting

Home / Add new posting

ZEROZERO
Outlet owner

- Dashboard
- My Postings
- Edit profile
- Reset password
- Add new posting
- My Adverts
- Add nearby locations
- Messages/Suggestions
- Logout

Add a posting

Please do a search before adding the post to verify whether the outlet to be added is already existing.

State
-- select state --

district
-- select district --

city
-- select city --

which category the outlet belongs to
-- select Category --

Name of the outlet
Outlet's name

My Ads

Home / My Ads

My Advertisements

Total ads 3 [Add new Advertisement](#)

Ad content	outlet name	Package	validity	expiring in	Status	Payment Status	Actions
	ZeroCorp pvt.	SILVER	10 days	10 days	ACTIVE	INCOMPLETE	Actions unavailable
	ZeroCorp pvt.	STANDARD	20 days	19 days	ACTIVE	COMPLETE	
	Arya	STANDARD	20 days	19 days	ACTIVE	COMPLETE	

Payment info

Home / My Ads

Payment info

Total payments 11 [Completed payments 11](#)

Payment id	Payer's name	Outlet name	card number	Amount	Payment Status
6	ZeroSlasher	ZeroCorp pvt.	6759649826438453	100	COMPLETE
7	ZeroSlasher	ZeroCorp pvt.	4048344905738876	100	COMPLETE
1	zeroZero	ZeroCorp pvt.	4048344905738876	100	COMPLETE
2	zeroZero	ZeroCorp pvt.	4048344905738876	300	COMPLETE
3	zeroZero	ZeroCorp pvt.	4048344905738876	300	COMPLETE
4	zeroZero	ZeroCorp pvt.	4048344905738876	200	COMPLETE
5	zeroZero	ZeroCorp pvt.	4048344905738876	200	COMPLETE

KNOWHERE HOME [My Account](#)

Profile Settings

Home / Profile Settings

USER
Administrator

- Dashboard
- Approve Outlets
- User Management
- Outlet Management
- Post Reportings
- Payment informations
- Messages/Enquiries
- Logout

Approve requests

Total Requests
4 Requests

Approved Requests
2 Approved

Pending approvals
2 Pending

Outlet name	City	Category	Owner name	Phone	Email	Proofs	Status	Actions		
ZeroCorp	kattappana Kottayam- KERALA	ATM	zero	8547880393	bigboystuffss@gmail.com				APPROVED	
choice	kattappana Kottayam- KERALA	vetenary	john	1234567890	usernotalive@gmail.com				APPROVED	
Revolt	kattappana Kottayam- KERALA	Clinic	william	1234567890	albinsalu@mca.ajce.in				DISAPPROVED	
blazer	ponkunnam Kottayam- KERALA	Bakery	shrike	1234567890	thomasjoseph@mca.ajce				DISAPPROVED	

KNOWHERE HOME ABOUT CONTACT [My Account](#)

Messages

Home / Messages

ZEROZERO
Outlet owner

- Dashboard
- My Postings
- Profile settings
- Reset password
- Add new posting
- Add nearby locations
- My Adverts
- Messages/Suggestions
- Logout

bigboystuffss@gmail.com
Subject: dlghj
unread

Chat Messages

bigboystuffss@gmail.com
fghjk

bigboystuffss@gmail.com
test response

Type Here & Press Enter

KNOWHERE HOME [My Account](#)

Profile Settings

Home / [Profile Settings](#)

- Dashboard
- Approve Outlets
- User Management**
- Outlet Management
- Post Reportings
- Payment informations
- Messages/Enquiries
- Logout

User Management

Total Users
5 users

Blocked users
0 users

Active users
5 users

Username	email	phone	Status	Actions
Fiend	fiend@gmail.com	6222345678	ACTIVE	
aaa	ALBINSALU24@GMAIL.CO	8520741963	ACTIVE	

KNOWHERE HOME ABOUT CONTACT [My Account](#)

Profile Settings

Home / [Profile Settings](#)

- Dashboard
- My Postings
- Profile Settings**
- Reset password
- Add new posting
- My Adverts
- Add nearby locations
- Messages/Suggestions
- Logout

Contact Detail

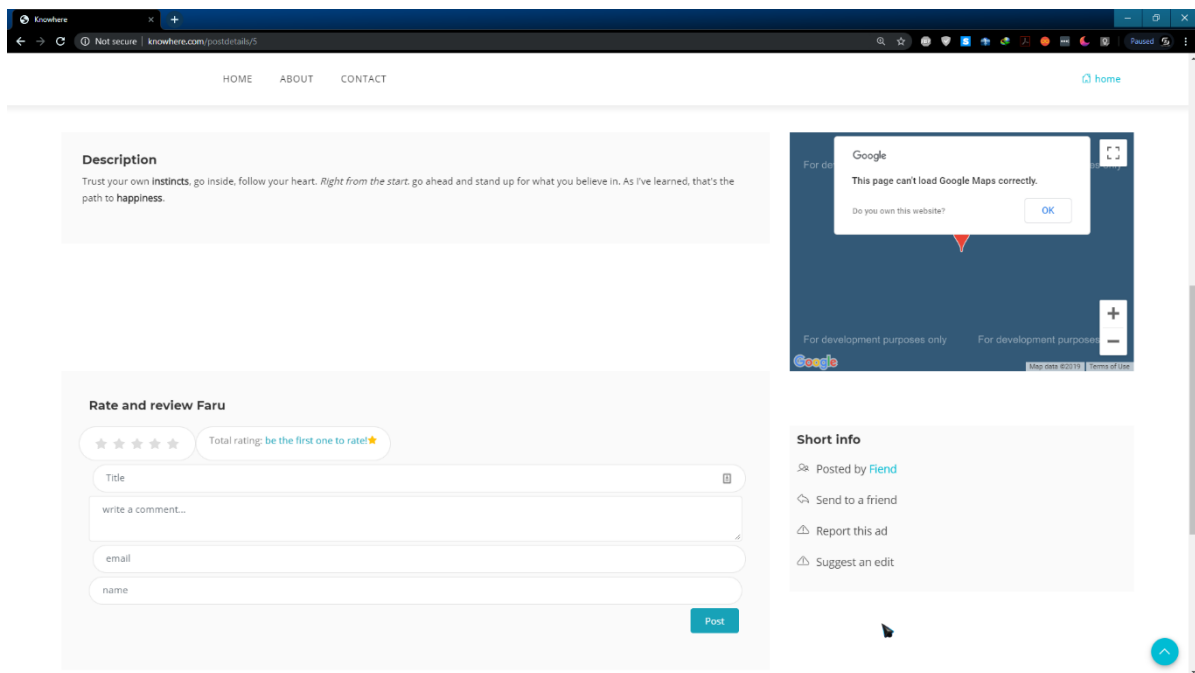
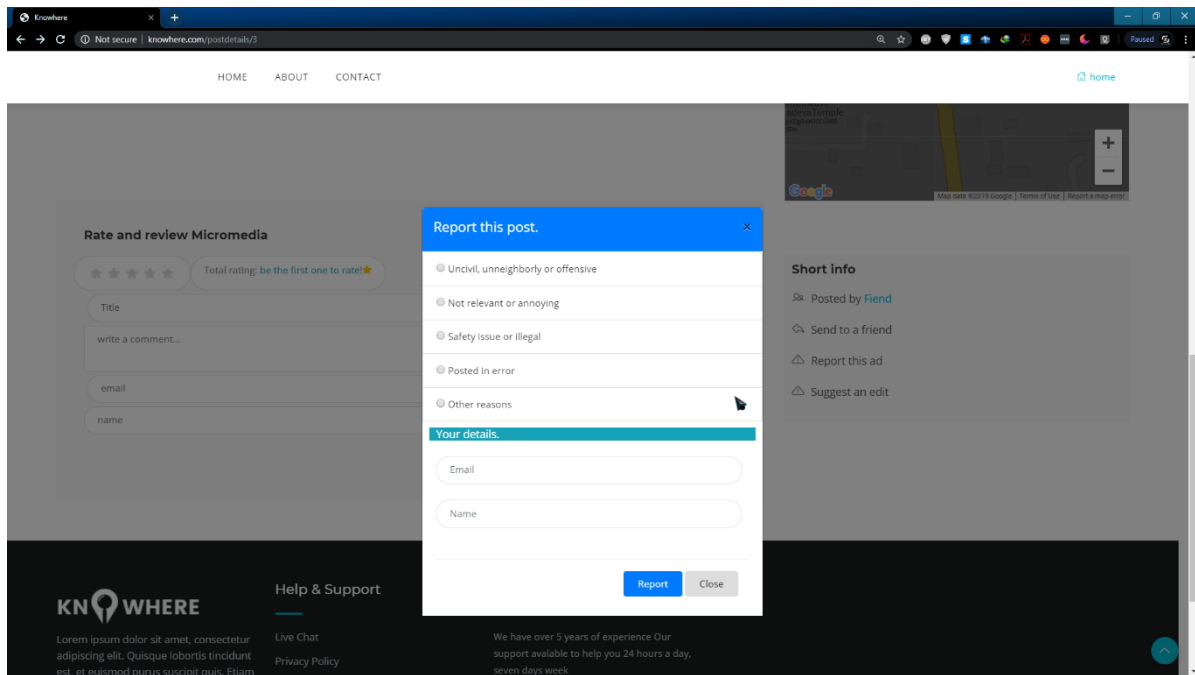
Name
zeroZero

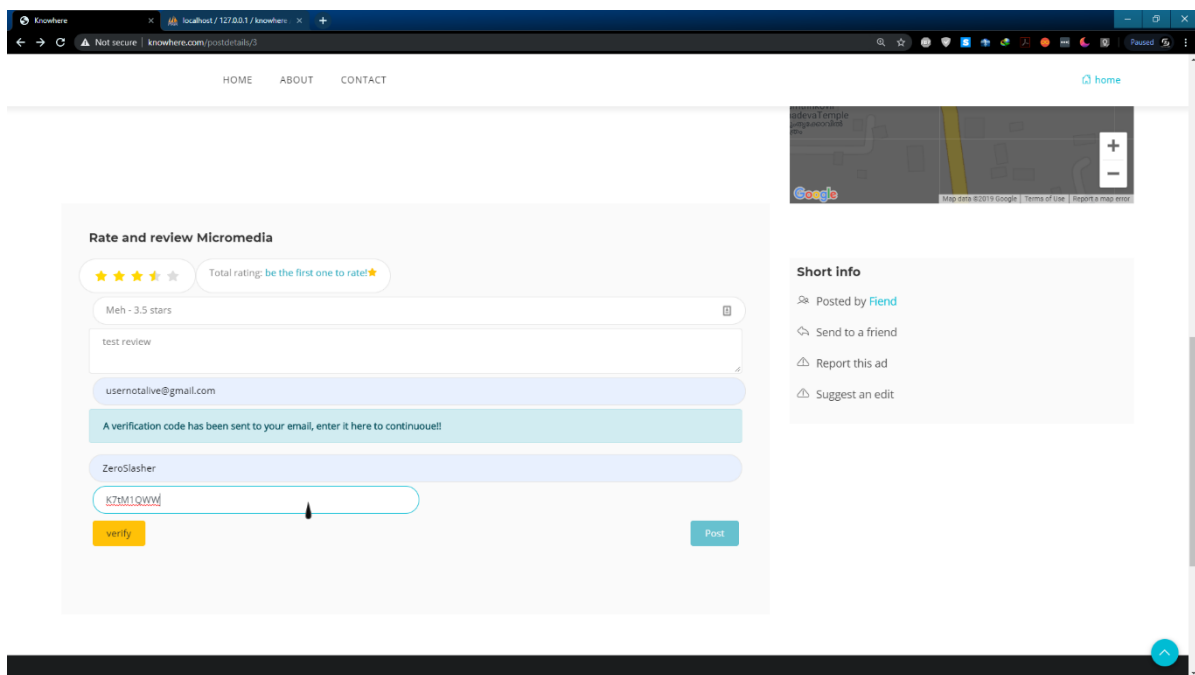
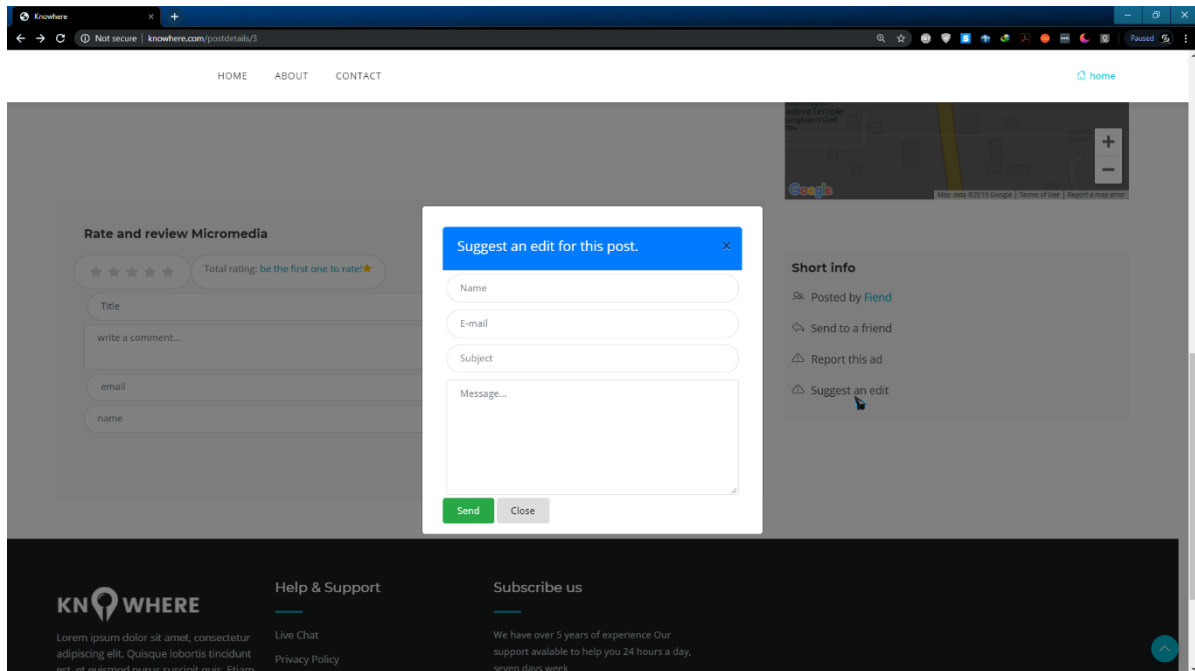
Title
You dont know me.. Youre about to!

Contact number
7410852096

Address
address

Add profile image
[Choose File](#) No file chosen





KNOWHERE HOME ABOUT CONTACT [My Account](#)

Payments

Home / My ads / Payments

Your Order

Product	Total
Posting Ad for ZeroCorp pvt.	
Payee zeroZero	
Package selected: SILVER	
Package amount: ₹ 100	
Total	₹ 100

Payment Method

Name on Card *
John doe

Credit Card Number
1234 5678 9012 3456

CVV* Expiry month* Expiry year*
123 MM YY

GRAND TOTAL: ₹100

[Complete payment Now](#)

[Help & Support](#) [Subscribe us](#)

KNOWHERE HOME ABOUT CONTACT [Home](#)

Forgot Password

Home / Forgot Password

Forgot Password

Email

[Send me my Password](#)

[Don't have an account?](#) [Back to Login](#)

[Help & Support](#) [Subscribe us](#)

KNOWHERE

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque lobortis tincidunt est, et euismod purus suscipit quis. Etiam euismod ornare elementum. Sed ex est, consectetur eget facilisis sed, auctor ut

Help & Support

[Live Chat](#)
[Privacy Policy](#)
[Purchase Protection](#)
[Support](#)

Subscribe us

We have over 5 years of experience Our support available to help you 24 hours a day, seven days week

Email [Send](#)

2.12.2 SAMPLE CODE

LoginController

```
<?php

namespace App\Http\Controllers;

use App\Login;
use DB;
use Hash;
use Illuminate\Http\Request;
use Session;

class LoginController extends Controller
{
    public function login(Request $request)
    {
        $email = $request->get('email');
        $hash = $request->get('password');
        //$password = Hash::make($hash);//make hash of plain text pwd

        //Hash::check($hash, $password);//for checking

        $a = Login::where('email', $email)->get();

        foreach ($a as $object) {

            $dbpwd_h = $object->password; //hash
```

```
$hash; //plain
```

```
if (Hash::check($hash, $dbpwd_h)) {
```

```
    $sutype = $object->utype_id;
```

```
    $semail = $object->email;
```

```
    $uid = $object->id;
```

```
if ($object->status_id == 2) {
```

```
    return redirect('/')->with('msg', 'User account temporarily blocked');
```

```
} else {
```

```
    if ($object->utype_id == '1' && $object->status_id == 1) {
```

```
        // session(['email'=>$semail]);
```

```
        // session(['utype'=>$sutype]);
```

```
        Session::put('id', $semail);
```

```
        Session::put('uid', $uid);
```

```
        Session::put('utype', $sutype);
```

```
        return redirect('/admindashboard');
```

```
    }
```

```
elseif ($object->utype_id == '2' && $object->status_id == 1) {
```

```
    Session::put('id', $semail);
```

```
    Session::put('uid', $uid);
```

```
    Session::put('utype', $sutype);
```

```
    $ownreg = DB::table('tbl_users_reg')->where('id',$uid)->get();
```

```
    foreach ($ownreg as $own) {
```

```
$name = $own->name;

}

Session::put('name', $name);

return redirect('/ownerdashboard');
} else if ($object->utype_id == '3' && $object->status_id == 1) {
    Session::put('id', $semail);
    Session::put('uid', $uid);
    Session::put('utype', $sutype);

    $usrreg = DB::table('tbl_users_reg')->where('id',$uid)->get();
    foreach ($usrreg as $usr) {
        $name = $usr->name;
    }
    Session::put('name', $name);
    return redirect('/userdashboard');
}
}
}

return redirect('/')->with('msg', 'incorrect login credentials');

}

public function logout()
{
```

```
//session_start();

//session_destroy();

Session::forget(['id', 'utype']);

Session::flush();

return redirect('/')->with('msg', 'Logged out successfully');

}

}
```

DashController

```
<?php

namespace App\Http\Controllers;

use DB;
use Session;

class DashController extends Controller
{
    public function admindash()
    {
        $utype = Session::get('utype');
        if (Session::get('id') && $utype == 1) {
            return view('admin_dashboard');

        } else {

            return redirect('/');

        }
    }
}
```

```
}

public function owndash()
{
    $utype = Session::get('utype');
    $uid = Session::get('uid');
    if (Session::get('id') && $utype == 2) {
        $udata = DB::table('tbl_users_reg')
            ->join('tbl_city', 'tbl_users_reg.city_id', '=', 'tbl_city.city_id')
            ->where('id', $uid)->get();

        return view('owner_dashboard', compact('udata'));
    } else {
        return redirect('/');
    }
}

public function userdash()
{
    $utype = Session::get('utype');
    $uid = Session::get('uid');

    if (Session::get('id') && $utype == 3) {
        $udata = DB::table('tbl_users_reg')->where('id', $uid)->get();

        foreach ($udata as $e) {
            $city_id = $e->city_id;

        }
    }
}
```

```
        if ($city_id == null) {  
            return view('user_dashboard', compact('udata'));  
        } else {  
  
            $udata = DB::table('tbl_users_reg')  
                ->join('tbl_city', 'tbl_users_reg.city_id', '=', 'tbl_city.city_id')  
                ->where('id', $uid)->get();  
  
            return view('user_dashboard', compact('udata'));  
        }  
    } else {  
        return redirect('/');  
    }  
}  
}
```

AdminController

```
<?php
```

```
namespace App\Http\Controllers;
```

```
use App>ListRqst;
```

```
use App\Reg;
```

```
use DB;
```

```
use Session;
```

```
class AdminController extends Controller
{
    public function approveoutlet()
    {
        $utype = Session::get('utype');
        if (Session::get('id') && $utype == 1) {
            $id = Session::get('id');
            $rqst = DB::select('SELECT * FROM `tbl_listing_rqst` as l, `tbl_city` as
c, `tbl_subcat` as s, `tbl_status` as st, `tbl_state` as sta, `tbl_district` as d WHERE
l.city_id = c.city_id AND l.subcat_id=s.subcat_id and l.status_id=st.status_id and
c.`dist_id`=d.`dist_id` and d.`state_id`=sta.state_id');
            $total = ListRqst::count();
            $pending = ListRqst::where('status_id', '=', 3)->count();
            $approved = ListRqst::where('status_id', '=', 4)->count();
            // return view('approve-outlet', ['rqst' => $rqst], ['pending' => $pending],
['total' => $total], ['approved' => $approved]);
            return view('approve-outlet', compact('approved', 'pending', 'total', 'rqst'));
        } else {
            return redirect('/');
        }
    }

    public function usermanagement()
    {
        $utype = Session::get('utype');
        $id = Session::get('id');
```

```
if (Session::get('id') && $utype == 1) {

    $urqst = DB::select('SELECT * FROM `tbl_users_reg` as l,`tbl_status` as
st,`tbl_login` as lo WHERE l.status_id=st.status_id and l.id = lo.id and lo.utype_id =
3 ');

    $total = Reg::count();

    $active = Reg::where('status_id', '=', 1)->count(); //return no. by checking
with utype_id

    $blocked = Reg::where('status_id', '=', 2)->count(); //return no. by checking
with utype_id

    // $blocked = DB::select('SELECT count(*) FROM tbl_user_reg where
status_id =2');

    return view('user-management', compact('urqst', 'total', 'blocked', 'active'));
} else {
    return redirect('/');
}
}

public function outletmanagement()
{
    $utype = Session::get('utype');

    if (Session::get('id') && $utype == 1) {

        $orqst = DB::select('SELECT * FROM `tbl_users_reg` as l, `tbl_city` as
c,`tbl_status` as st, `tbl_login` as lg WHERE l.city_id = c.city_id and
l.status_id=st.status_id and l.id=lg.id and lg.utype_id = 2');

        $total = Reg::count();

        $active = Reg::where('status_id', '=', 1)->count();
```

```
$blocked = Reg::where('status_id', '=', 2)->count();

return view('outlet-management', compact('orqst', 'total', 'blocked',

'active'));

    } else {
        return redirect('/');
    }
}

public function paymentinfo()
{
    $payment = DB::table('tbl_payments')
        ->join('tbl_users_reg', 'tbl_payments.id', '=', 'tbl_users_reg.id')
        ->join('tbl_advert', 'tbl_advert.ad_id', '=', 'tbl_payments.ad_id')
        ->join('tbl_outlet_prof', 'tbl_advert.outletid', '=', 'tbl_outlet_prof.outletid')
        ->join('tbl_status', 'tbl_payments.status_id', '=', 'tbl_status.status_id')
        ->get();

    $comp = DB::table('tbl_payments')->where('status_id', 10)->get();
    $c_comp = count($comp);
    $total = count($payment);
    return view('paymentinfo', compact('payment', 'total', 'c_comp'));
}
}
```