1. Download composer from getcomposer.org
   1. Open the downloaded exe file
   2. If php is set to global then we can download composer from command prompt with commands given on the getcomposer.org for terminal download.
   3. After installing composer you can type composer –v or just composer to see that composer is installed or not.
2. Now install git from git-scm.com
   1. And check after installation if git is installed or not by typing git in the CMD
   2. Now install laravel from laravel .com
   3. **Two methods to install laravel.** 
      1. **Via laravel** which is to install laravel first using composer.
         1. Through laravel we install first laravel globally
         2. And then we **don’t have to specify any git repo** but just write command laravel new and then directory name and the laravel project will be installed on the mentioned directory in the current directory you are working in cmd.
      2. **Via composer** by issuing the composer command create-project.
         1. **Installing via composer.**
         2. In CMD go to htdocs folder through CD.
         3. And type command to install laravell project.
         4. Composer create project-project laravel/laravel azamstlaravel.
         5. Laravel/laravel is git repository from the composer will download laravel for us.
         6. And azamstlaravel will be the folder name which will be made automatically.
         7. It will install all required dependencies itself.
   4. **Both methods** are much similar not that much different.
3. Now **when starting the project we give command (php artisan serve)** which will give a url for that project
   1. We can **close the project through** ( **control + c** )
   2. **Not starting project through php artisan serve**. We can **locate to the project directory** and then **go to public folder** then it will show us the project main page.
   3. **Our entry point in laravel project is index.php which is located in public folder** all our files are accessed through that index file.
4. After this the **second entry point is web.php located in routes is responsible for** showing the welcome to laravel page.
5. When open the web.php and see an anonymous function which routed in this file and this function calls a welcome page and view it that’s how it is working
6. In laravel / slash means root folder.
7. So they specify that whenever a person comes to root folder means the slash the anonymous function will show them the welcome page. it will load the view (welcome file)
8. So where is view locate : (in resources/views there will be view file with name of welcome.blade.php)
9. Welcome.blade.php is equivalent to simple welcome
10. Laravel include blade.php automatically with your views.
11. If we open the welcome.blade.php we will see some different tags like @auth @else etc. so these are blade templating engine which Is provided by laravel
12. Now make a new page about.blade.php don’t forget to mention blade.php when creating views.

And link it in welcome page which we can just write about in the href because laravell include the root URL automatically.

1. This is how we route views through anonymous function.
2. So in web.php route is the base class which have two static methods get and post static can be accessed through double colon:: we will use get to get data and post to submit data.
3. We can also return a text in anonymous function
4. And we can pass parameter to anonymous function as a variable for that we will specify in route get function first parameter that this function should expect parameters syntax for this is. Write in curly bracket the variable name like this {id} e.g after about {id}.
5. **( REMOVE PUBLIC FROM URL )**

We can remove public from URL for uploading to server by copying the server.php file in the root directory and pasting it as an index.php and also we will copy .htaccess file from public to the root folder and then it should work without calling php artisan serve.

1. **(how to handle views)**

How to pass a parameter to view from web.php routes

First we create view

Then we return view function in the anonymous function where we returned a value or text.

Syntax : return view(‘viewpagename’,[‘user’=>$user]);

Here in the view function in first parameter we passed the view file name which should be called for this url and in the second parameter we passed value or array to a view which can be used in view to show

1. **(Controller)**

Controllers can be found in the following directory in laravel

* App/http/controller
* We can create controller using artisan command line through the following command
* Php artisan make:controller ControllerName
* We call controller on routes by removing anonymous function and writing the class name and after that we use @ and function name in the controller which we want to call.
* Syntax : Route::get('user/{id}','MainPageController@showuser');

1. **(migrates)**

* The rule for migration in database is that the table name should be in small letters and it will be plural.
* When we are adding index to table through migration we will have to name it as the model name underscore id which will be usefull in automatic relationships. E.g contact\_id

1. **(model)**

* Command for making model : Php artisan make:model modelname –m
* This –m in the last will create migration for the model we are going to create and this migration will have two fields by default.
* Model name should be singular always and the first character will be capital.

1. **(Getting data from database)**

* We can get data from database using eloquent model for the table.
* We call that model in our controller and we can get data in many ways.

1. **(relationships between tables through eloquent models)**

* First we will create migration with schema e.g
* Php artisan make:migration create\_users\_table –create=users
* Then we will make another migration which we want to make relationship with
* And then we will add attributes to these tables e.g
* In user table we will specify role table id just like we make normal attribute
* Now laravel will automatically make relationship between these tables
* Rules to remember : when specifying another table attribute for relation ship it should be written like this: the table name should be same but singular when making relationship

Tablename\_attributename == user\_id

If you did not specify it correctly laravel will not recognize it.

* After this insert values based on relation through phpmyadmin
* Then we will create controller for each table we know how to create controller
* But we will write –r at the end of making controller command for making resources in the controller.
* (resource controller)

It is a controller which have built in methods and we don’t have to write methods in it as important methods are already available. E.g

Php artisan make:controller UserController –r

* Then we make model for each migration and in model we define relationships please see this documentation for better understanding.

<https://laravel.com/docs/5.8/eloquent-relationships#introduction>

* Now fetch all data through usermodel and send data to views.
* And then config in routes web.php
* In web.php we route with controller name for normal controllers but we have made resources controller which have different way to route.
* E.g we define = Route::resource('roles','RoleController');

1. **{{asset(‘public/js/custom.js’)}}**

We can use asset function to call or include css and js files or images in our templets. We will have to place our files in the public folder which is located in root directory.

**Note: Action, route, and other helper functions are available in laravel docs search helpers and you can find a lot of helpers’ functions for every kind of work.**

1. **Tinker CLI?**

* Through tinker we can fill database tables with random
* First we have to put details of a table migration in userfactory file located in databases factory.
* Then in cmd we write php artisan tinker
* And then the tinker starts then we put command like below
* Factory(App\User,10)->create;
* This will create 10 records in user table

1. Asd
2. Asd
3. Asd
4. Asd
5. Asd
6. Asd
7. Asd
8. Asd
9. Asd
10. Asd
11. Asd
12. Asd
13. Asd
14. Asd
15. Asd
16. Asd
17. Asd
18. Asd
19. **GIT Commands :**
    1. Uploading file to repository on git
       1. First create a repository on github
       2. Copy the url of your repository
       3. Open git bash
       4. Go to the folder which have the project you want to upload
       5. **git init** is a command that create a file that enables you to upload file in the GitHub repo.
       6. Git add . command add all the files in a temporary folder for upload
       7. Git commit –m ‘message’ to commit files
       8. Git remote add origin url command for adding repo link
       9. Git push –u origin master
       10. Might ask you username and password when pushing as a master.