

ADVANCED WEB TECHNOLOGY LABORATORY
PRACTICAL FILE
(LPEIT-102)



Submitted by:
Chandan Kumar
Dept - D3 IT A1
C.R.N: 2021022
U.R.N: 2004899

Submitted to:
Pf. Jagdeep Singh

Department of Information Technology
Guru Nanak Dev Engineering College
Ludhiana(PUNJAB)

Index

1. Bootstrap Installation.....	2
2. Bootstrap Grid Layout.....	5
3. Navbars,Modals,Buttons and Progress Bars.....	6
4. To create and setup Git Repository on Github using SSH.....	8
5. To perform push,clone and patch operations on a git repository.....	14
6. To install and setup the codeigniter framework and to understand its MVC Architecture.....	17
7. To construct login page using codeigniter framework and also perform CRUD operations.....	20
8. To install , setup and configure the Laravel Framework.....	24
9. To construct any simple web application using Laravel Framework.....	28

Practical No. 1

To install and setup the HTML5 based bootstrap framework and to deploy the basic HTML elements using bootstrap CSS.

Bootstrap: It is a powerful, feature-packed frontend toolkit that can be used to build anything—from prototype to production—in minutes. It has inbuilt and pre designed codes for quickly creating basic layout of any kind of website.

Installation and Set up:

1. Go to the official bootstrap website i.e <https://getbootstrap.com/> and click on the Download option.

Download

Bootstrap (currently v3.3.7) has a few easy ways to quickly get started, each one appealing to a different skill level and use case. Read through to see what suits your particular needs.

Bootstrap

Compiled and minified CSS, JavaScript, and fonts. No docs or original source files are included.

Download Bootstrap

Source code

Source Less, JavaScript, and font files, along with our docs. **Requires a Less compiler and some setup.**

Download source

Sass

Bootstrap ported from Less to Sass for easy inclusion in Rails, Compass, or Sass-only projects.

Download Sass

Fig 1.1: Bootstrap official website.

2. After clicking on the download button , all the bootstrap files will be downloaded as a zip file.

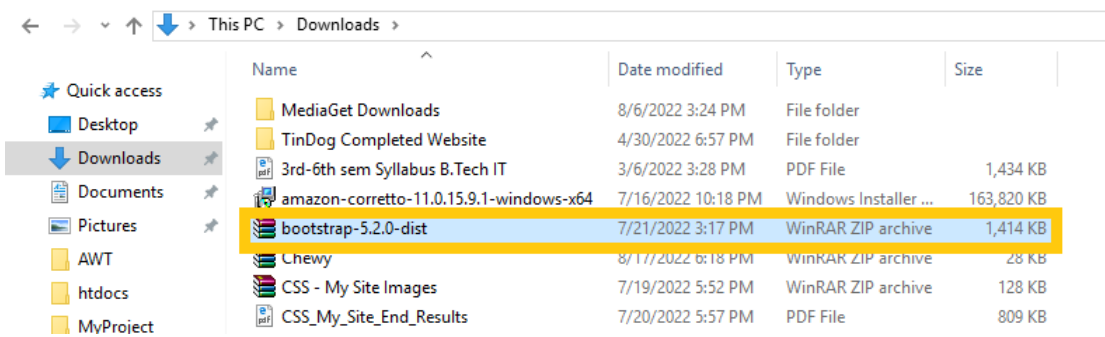


Fig 1.2: Downloaded bootstrap zip files.

3. After finding the downloaded files, double click and extract two folders named css and js.

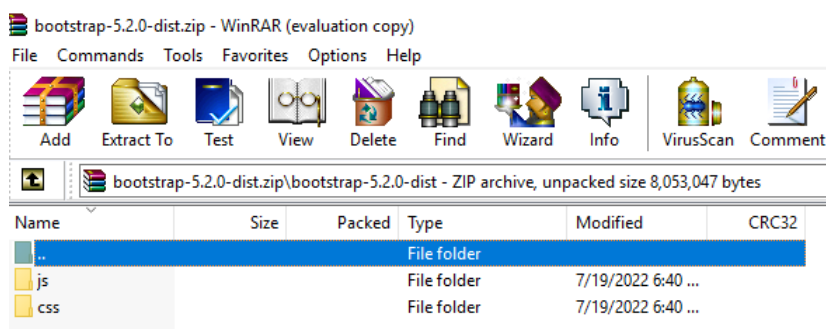


Fig 1.3: Extracting the zip file containing css and js folder.

4. Now you can move these folders to your HTML files and start using them by attaching their links.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>frontpage</title>
  <link rel="stylesheet" href="bootstrap-5.2.1-dist/css/bootstrap.css" />
  <link rel="stylesheet" href="style.css">
  <script src="bootstrap-5.2.1-dist/js/bootstrap.js"></script>
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.2.0/css/all.min.css">
```

Fig 1.4 :Including bootstrap 5 into the project.

Deploying Basic Elements:

- **Navbar** : A navigation bar helps readers in selecting topics, links or sub-topics of their interest. Using a navigation bar, users need not enter the URL of the specific webpage, as this is automatically taken care of by the navigation bar, with the navigation sections having embedded the necessary links of the webpage.
- **Drop down menus** : It is a toggleable, contextual menu for displaying lists of links which can be made interactive with the dropdown JavaScript plugin.
- **Cards**: It is a flexible and extensible content container. It includes options for headers and footers, a wide variety of content, contextual background colors, and powerful display options. If you're familiar with Bootstrap 3, cards replace our old panels, wells, and thumbnails.
- **Buttons** : Bootstrap includes several predefined button styles, each serving its own semantic purpose, with a few extras thrown in for more control. They can be used for custom button styles for actions in forms, dialogs, and more with support for multiple sizes and states.
- **Carousel**: It is a slideshow for cycling through a series of content, built with CSS 3D transforms and a bit of JavaScript. It works with a series of images, text, or custom markup. It also includes support for previous/next controls and indicators.

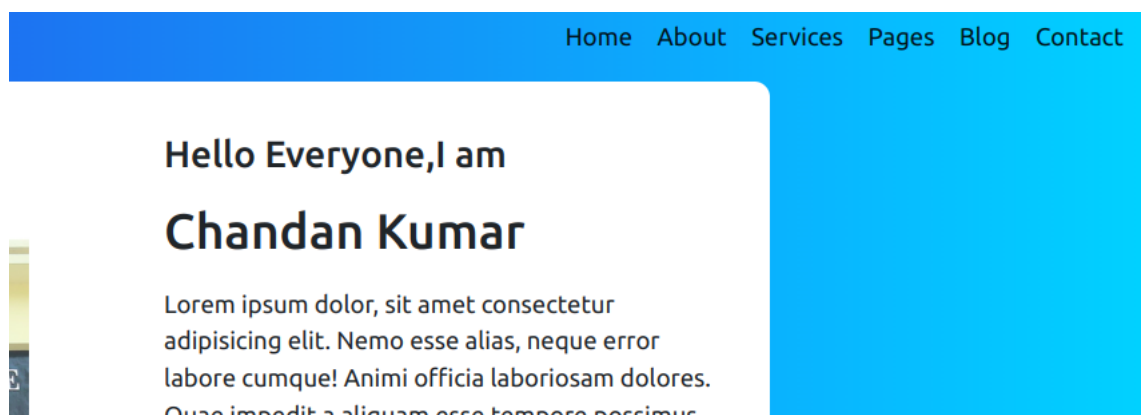


Fig 1.5: Deploying basic bootstrap elements.

Practical No. 2

To understand and deploy the multicolumn grid layout of bootstrap.

Bootstrap's grid system uses a series of containers, rows, and columns to layout and align content. It's built with flexbox and is fully responsive. Bootstrap Grid System is a collection of HTML and CSS constituents that enables you to roll out a website and helps you in keeping its content at significant settings with ease.

The basic components of the grid system are -

- **Container:** A div element effectively holds rows and columns and plays a vital role in offering the correct width to a particular layout. We can provide specific classes to it for layout. Container element wraps up the entire grid. In case we need the container to cover the full width, we can use the container-fluid class.
- **Rows:** It is a class given to the div element. This row component basically covers the horizontal component of the grid that discharges the padding menu and acts as a wrapper all over the columns.
- **Columns:** In Bootstrap, different column class prefixes are used for the diverse sizes of devices. Basically we can divide our row into a maximum of 12 column layout using this class.

Demonstrating basic grid layout														
col-4				col-4				col-4						
col-6						col-6								
col-3			col-3			col-3			col-3					
col-6						col-3			col-3					
col-2		col-4				col-6								

Fig 2.1: Demonstrating the grid layout in bootstrap.

Practical No.3

To deploy different types of buttons, progress bars, navigation bars and modals using bootstrap.

- **Navigation Bar** helps readers in selecting topics, links or sub-topics of their interest. Using a navigation bar, users need not enter the URL of the specific webpage, as this is automatically taken care of by attaching the links to those pages or sections. When the user clicks their, they are henceforth directed to that specific section making it easy for a user to scroll through various sections.

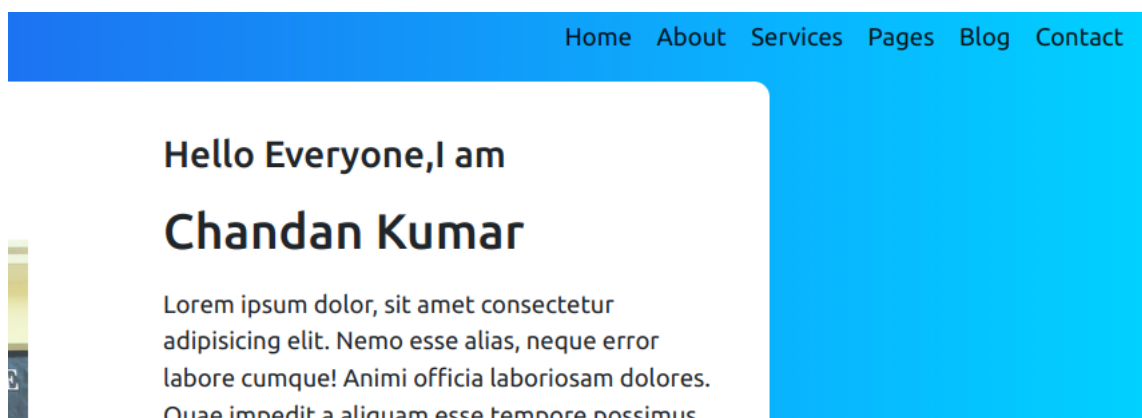


Fig 3.1: A basic navigation bar in bootstrap

- **Modals** are built with HTML, CSS, and JavaScript. They're positioned over everything else in the document and scroll from the body section so that content scrolls instead. For instance, here we have a registration modal, which on user click scrolls and asks for the registration details to the website.
- **Buttons** : We can create buttons of different sizes and colors using predefined bootstrap classes. There are many different pre-defined styles available for designing a button, all you need is to assign it the specific classes. For instance, primary creates a basic blue button, danger creates a button in red and so on. We can also create transparent and disabled buttons by giving them these specific classes.
- **Progress Bars** are usually used to show or highlight the progress of any kind of task being done. For instance if we want some kind of form filling as input from the user, we can use a progress bar there. These are built with two HTML elements, some CSS to set the width, and a few attributes.

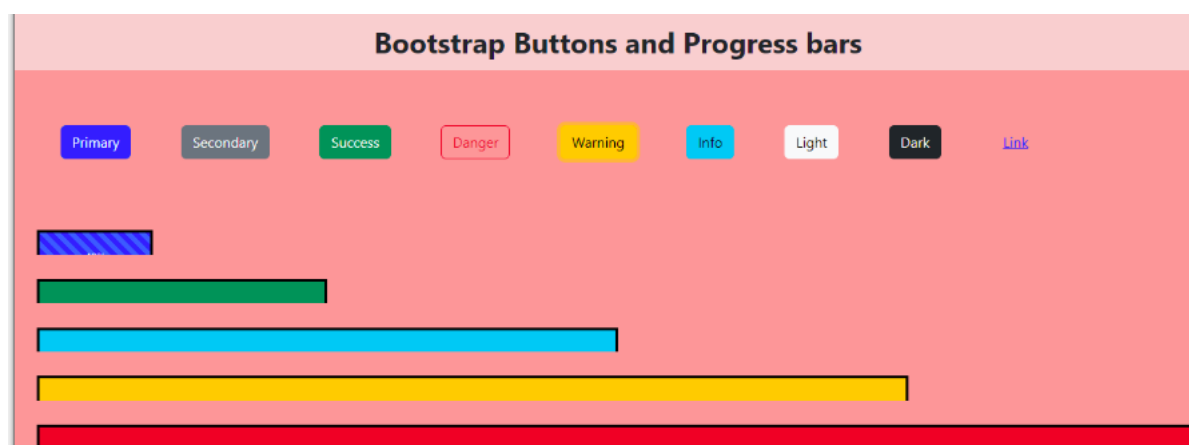


Fig 3.3: Bootstrap buttons and Progress Bars

Practical No.4

To create and setup the Git Repository on Github/Bitbucket using SSH.

GIT is a free and open source software for distributed version control: tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development.

Features of Git:

- Tracks history of projects.
- Free and open source software
- Supports non-linear development
- Creates backups for recovering lost data.
- Scalable
- Supports collaboration
- Branching is easier
- Distributed development.

Download and Installation :

- Go to the official website <https://git-scm.com/downloads> and download the package after selecting the suitable options. Install as any other normal software.

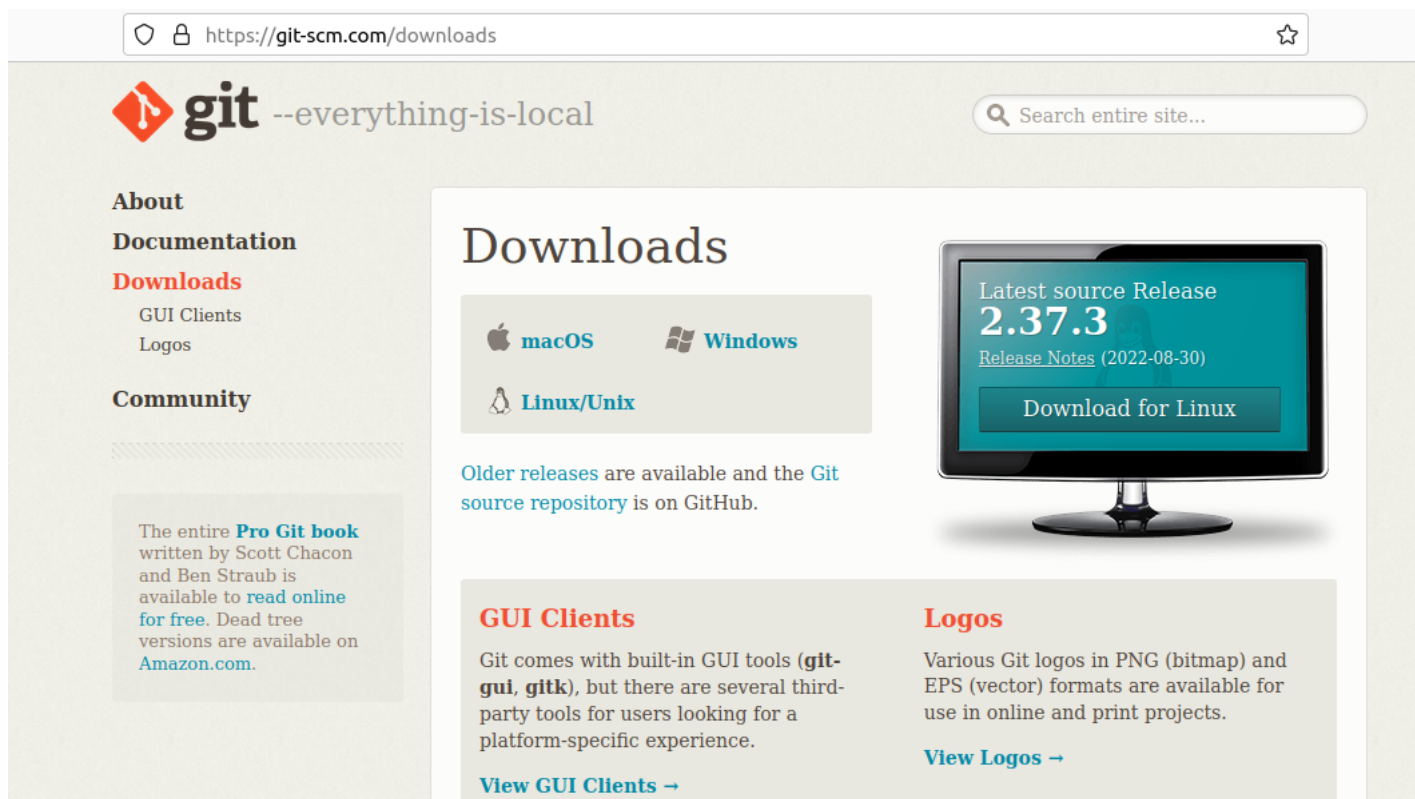


Fig 4.1: Downloading the git package

- You can check if the latest version has been installed by typing `git --version` on the terminal.

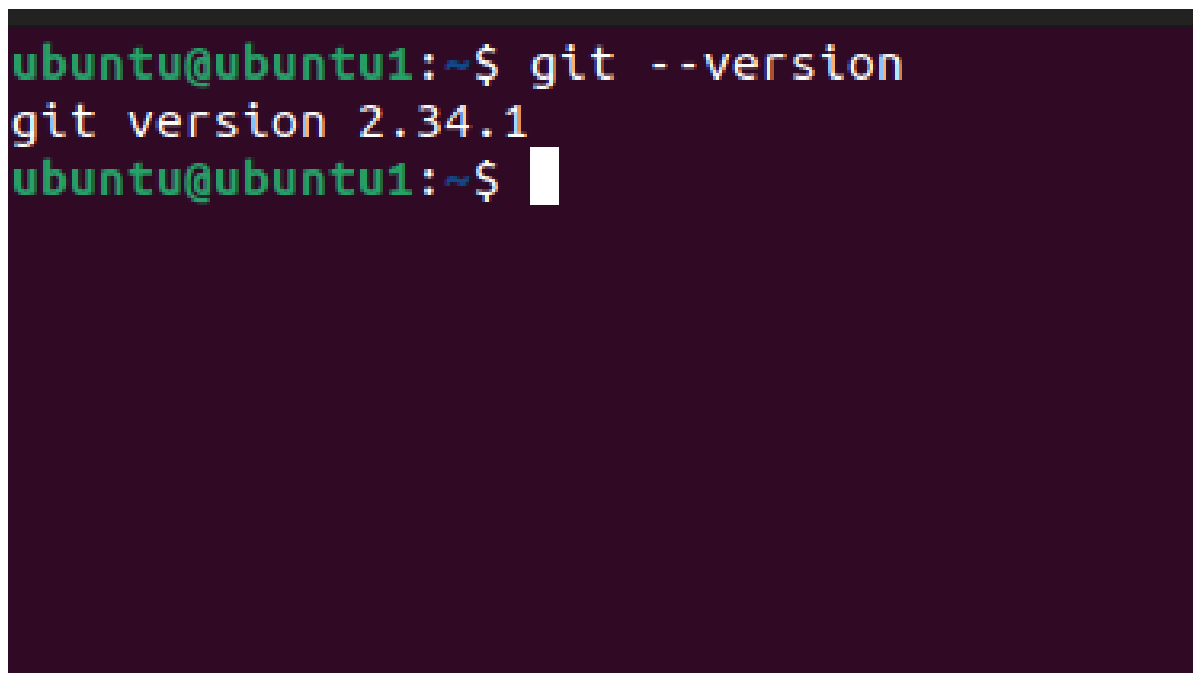


Fig 4.2: Verifying the Installation

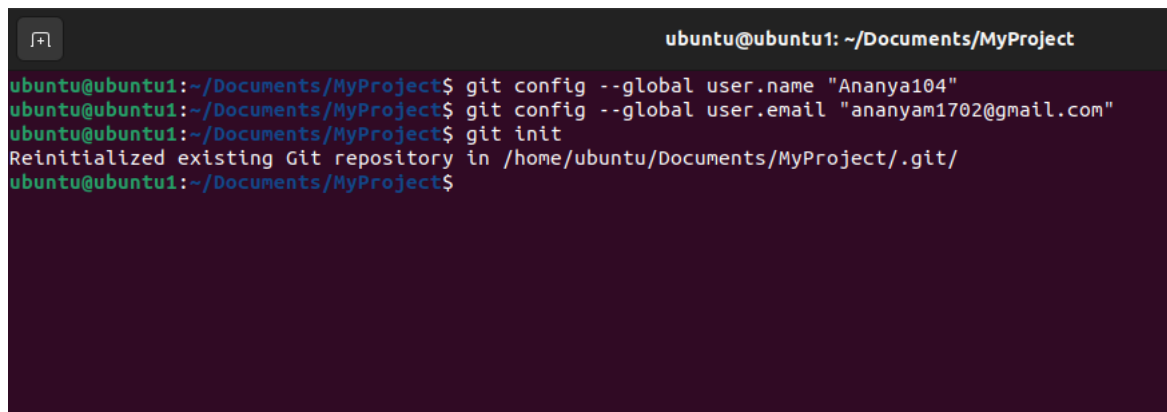
- After you are done with the verification , you need to configure git. Type the following commands in terminal:

```
git config --global user.name "Your github username"  
git config --global user.email "your email address."
```

With this, you are set to initialize git. Change directory to the required folder and use command :

```
git init
```

This will initialize an empty git repository.

A terminal window with a dark background and light green text. The title bar at the top reads 'ubuntu@ubuntu1: ~/Documents/MyProject'. The terminal shows the following commands and output:

```
ubuntu@ubuntu1:~/Documents/MyProject$ git config --global user.name "Ananya104"  
ubuntu@ubuntu1:~/Documents/MyProject$ git config --global user.email "ananyam1702@gmail.com"  
ubuntu@ubuntu1:~/Documents/MyProject$ git init  
Reinitialized existing Git repository in /home/ubuntu/Documents/MyProject/.git/  
ubuntu@ubuntu1:~/Documents/MyProject$
```

Fig 4.3: Configuring and Initializing.

- **Local Git:** Use the following commands -
git add . //adds all files present to the git repository
git add filename //adds the specific file to repository.
git status //tracks modified files.
git commit -m "your message."

```
ubuntu@ubuntu1:~/Documents/MyProject$ git add .
ubuntu@ubuntu1:~/Documents/MyProject$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
ubuntu@ubuntu1:~/Documents/MyProject$ git commit -m "first commit."
```

Fig 4.4: Adding files to local repository and committing changes.

- **Remote Git:**

- Adding a Remote Repository: To add a new remote, use the git remote add command on the terminal, in the directory your repository is stored at.

The git remote add command takes two arguments:

1. A remote name, for example, origin
 2. A remote URL
- The URL in the previous set can be added through two ways- SSH and https. Here we shall configure it using SSH I,e secure shell.

So, to set the URL , Use the following command -

```
git remote set-url origin git@github.com:USERNAME/REPOSITORY.git
```

After this,you can type :

git remote -v to check for the existing remotes.

```
ubuntu@ubuntu1:~/Documents/My Project$ git remote add origin git@github.com:Ananya104/Dog-Nation.git
ubuntu@ubuntu1:~/Documents/My Project$ git remote -v
origin  git@github.com:Ananya104/Dog-Nation.git (fetch)
origin  git@github.com:Ananya104/Dog-Nation.git (push)
ubuntu@ubuntu1:~/Documents/My Project$
```

Fig 4.5: Adding a new remote and setting url.

- Generating a SSH Key pair : Type the command -

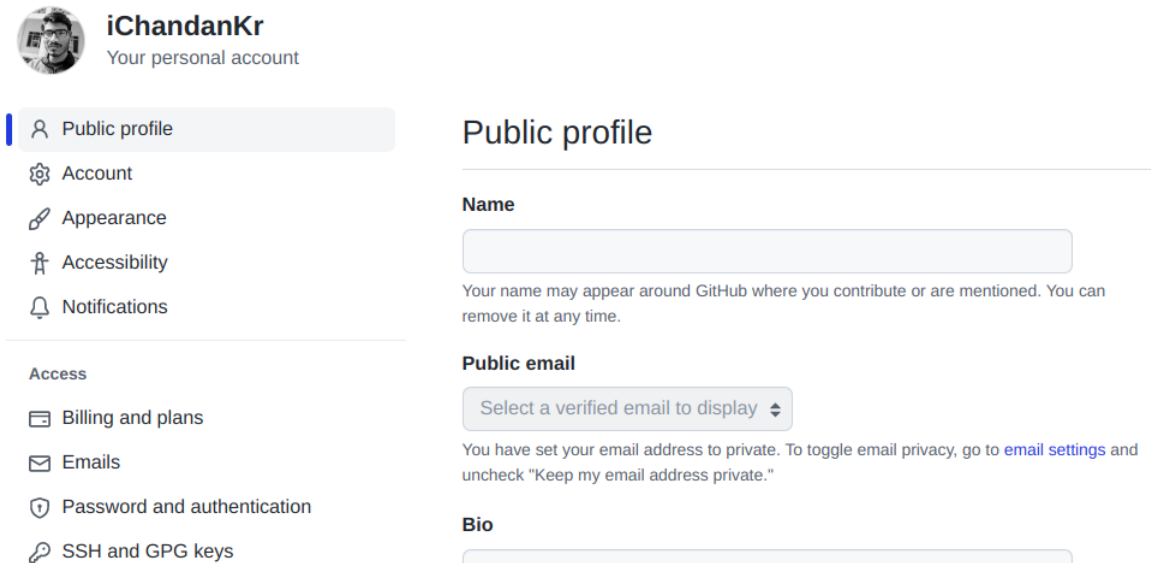
ssh-keygen -C "your email address."

Set up a passphrase and it will generate a key for you.

```
ubuntu@ubuntu1:~/Documents/My Project$ ssh-keygen -C "ananyam1702@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match. Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:JyLs3WVLSbrqIDH673pxpsTp4TZ6qy0dL1Em86ftBAM ananyam1702@gmail.com
The key's randomart image is:
+----[RSA 3072]-----+
|
|      .
|     o .
|    E . +
|   . o.o +
|  =..BS =
| o =0o+O.
| . o=oXo+.
| ..o@oo..
| =@=+...
+----[SHA256]-----+
ubuntu@ubuntu1:~/Documents/My Project$
```

Fig 4.6: Generating a new SSH key pair

- This command produces the two keys needed for SSH authentication: your private key and the public key. Now, we add this key generated to our github account.
- Go to your github account - Settings. There you will find the option SSH and GPG Keys. Click on it.



iChandanKr
Your personal account

Public profile

Name

Your name may appear around GitHub where you contribute or are mentioned. You can remove it at any time.

Public email

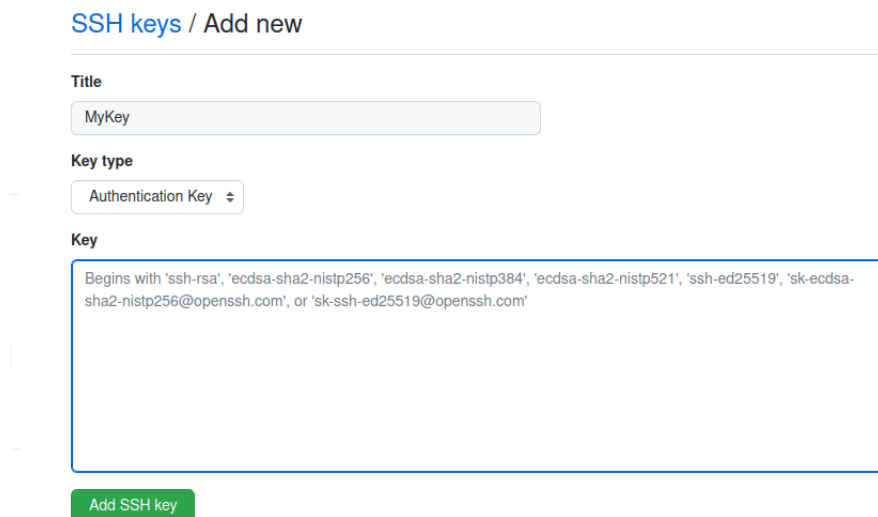
Select a verified email to display

You have set your email address to private. To toggle email privacy, go to [email settings](#) and uncheck "Keep my email address private."

Bio

Fig 4.7: Adding generated key to github

- After you click on SSH and GPG Keys, a window appears. Fill in a suitable title and add the key pair.



SSH keys / Add new

Title

MyKey

Key type

Authentication Key

Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

Add SSH key

Fig 4.8: Adding key to ssh agent

Now, we are done with the authentication. We can henceforth push our project to this remote repository by giving the set passphrase. This completes our ssh setup.

Practical No.5

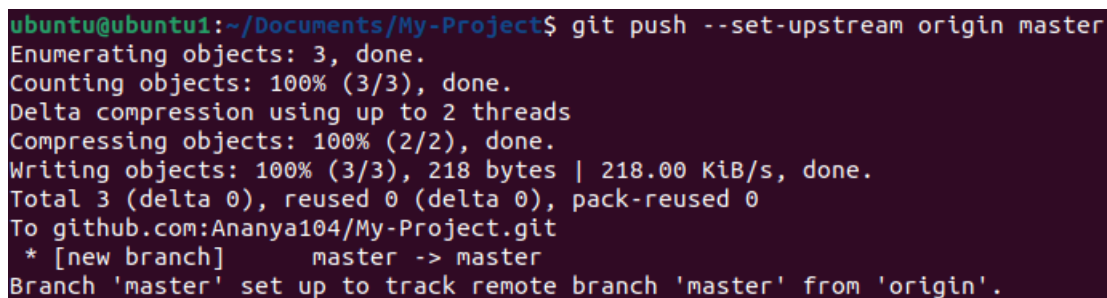
To perform push, clone and patch operation in git repository.

Let us take a look at all the three operations with their commands and use.

- **Git Push:** The git push command is used to upload local repository content to a remote repository. Pushing is how you transfer commits from your local repository to a remote repo. It's the counterpart to git fetch, but whereas fetching imports commits to local branches, pushing exports commits to remote branches. Remote branches are configured using the git remote command. Pushing has the potential to overwrite changes, caution should be taken when pushing.

To push your changes to the remote repository, we use the following command -

git push remote branch.



```
ubuntu@ubuntu1:~/Documents/My-Project$ git push --set-upstream origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 218 bytes | 218.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Ananya104/My-Project.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Fig 5.1: Pushing changes to remote repository

- **Git clone:** It is a Git command line utility which is used to target an existing repository and create a clone, or copy of the target repository. Cloning is generally a one-time operation. Once a developer has obtained a working copy, all version control operations and collaborations are managed through their local repository.

To make a clone of your repository, you need to type in the following command-

git clone url

[Here, we can clone using both https or SSH.]



```
chandan@kumar:~$ git clone https://github.com/iChandanKr/crud_final.git
Cloning into 'crud_final'...
remote: Enumerating objects: 810, done.
remote: Counting objects: 100% (810/810), done.
remote: Compressing objects: 100% (571/571), done.
remote: Total 810 (delta 205), reused 810 (delta 205), pack-reused 0
Receiving objects: 100% (810/810), 1.45 MiB | 1.94 MiB/s, done.
Resolving deltas: 100% (205/205), done.
```

Fig 5.2: Making a clone of remote repository

- **Patches:** git patch or git diff is used to share the changes made by you to others without pushing it to main branch of the repository. This way other people can check your changes from the GIT patch file you made and suggest the necessary corrections. After making all the corrections you can push the changes to main branch of the repository. Patch is a text file, whose contents are similar to Git diff, but along with code, it also has metadata about commits; e.g., commit ID, date, commit message, etc. We can create a patch from commits and other people can apply them to their repository.

- To create a git patch, firstly we need to make some changes/alterations in our code, add them and commit them. After this, we need to checkout to the other branch with which we are comparing it. Type the following command -

git format-patch branchname

This generates patch files as per commits and these are stored in a folder named patch.


```
ubuntu@ubuntu1:~/Documents/My-Project$ nano testfile1
ubuntu@ubuntu1:~/Documents/My-Project$ git status
On branch branch1
Your branch is up to date with 'origin/branch1'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   testfile1

no changes added to commit (use "git add" and/or "git commit -a")
ubuntu@ubuntu1:~/Documents/My-Project$ git add .
ubuntu@ubuntu1:~/Documents/My-Project$ git commit -m "1"
[branch1 df8b7f8] 1
 1 file changed, 1 insertion(+)
ubuntu@ubuntu1:~/Documents/My-Project$ git status
On branch branch1
Your branch is ahead of 'origin/branch1' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
ubuntu@ubuntu1:~/Documents/My-Project$ git format-patch master
0001-1.patch
ubuntu@ubuntu1:~/Documents/My-Project$ ls
0001-1.patch  testfile1  testfile2
```

Fig 5.3: Making a patch file.

After it is created, we can apply the patch file by using the command -
git am 000101.patches

This applies the patch and makes the required changes to the branch.

Practical No.6

To install and setup the codeigniter framework and to understand its MVC architecture.

CodeIgniter is a powerful PHP framework with a very small footprint, built for developers who need a simple and elegant toolkit to create full-featured web applications. CodeIgniter was created by EllisLab, and is now a project of the British Columbia Institute of Technology.

The framework can have core parts easily extended or completely replaced to make the system work the way you need it to. In short, CodeIgniter is the malleable framework that tries to provide the tools you need while staying out of the way.

Downloading and Setup :

- Go to the official website <https://codeigniter.com/> and click on "Download" option. We are basically following manual installation process here.

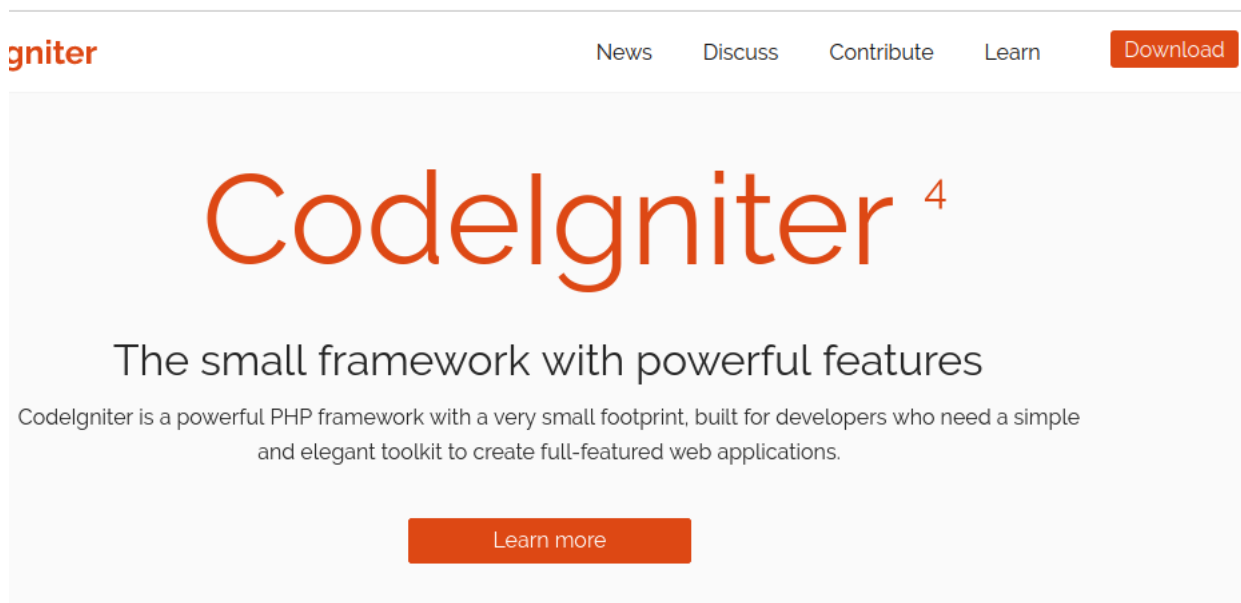


Fig 6.1: Downloading the package.

- The file is downloaded in the form of a zip file. Extract it at the root directory and rename it to CI.

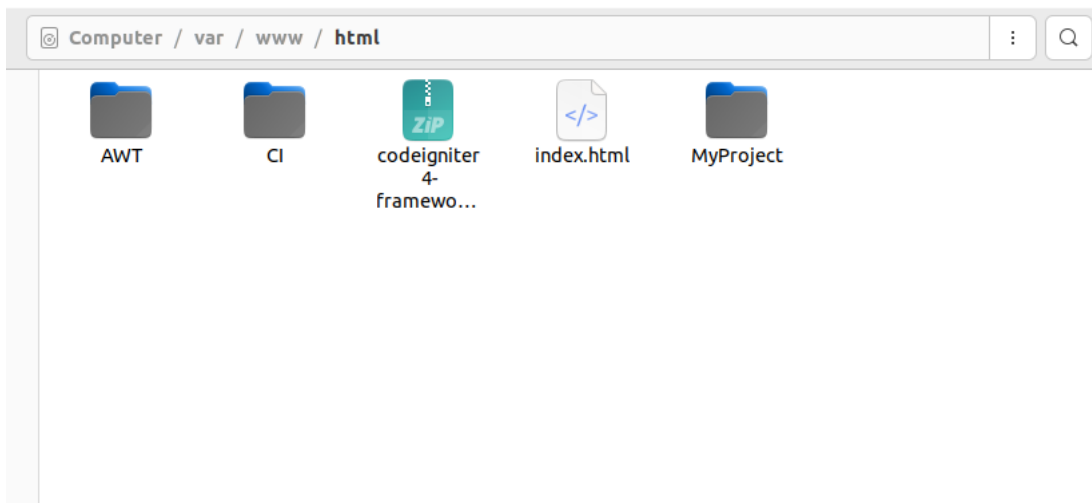


Fig 6.2: Extracting the package.

With this , our download and installation process is complete. Now we need to set up the framework.

- **Server requirements:** The first step is to install the various extensions required.

We need the following five extensions installed in order to be able to run our application.

1. php-json
2. php-intl
3. php-mysqlnd
4. php-xml
5. php-libcurl

- **Changing the base URL:** Open the app/Config/App.php file with a text editor and set your base URL as follows :

```
public $baseURL = http://localhost/CI/public/
```

```

24  "
25  * @var string
26  */
27  public $baseURL = 'http://localhost/CI/public/';
28
29  /**
30  * -----
31  * Index File
32  * -----
33  *
34  * Typically this will be your index.php file, unless you've renamed it to
35  * something else. If you are using mod_rewrite to remove the page set this
36  * variable so that it is blank.
37  *

```

PHI

Fig 6.3: Changing the base URL

- **Changing mode of writable directory :** In order to change the mode i.e the way in which the files can be accessed, we need the following command-

`chmod -R 777 writable/`

```
ubuntu@ubuntu1:/var/www/html/CI$ ls
app          LICENSE      phpunit.xml.dist  README.md  system  writable
composer.json  php.ini     public            spark      tests
ubuntu@ubuntu1:/var/www/html/CI$
```

Fig 6.4: Changing the access mode.

- After this we are set to display the first page, i.e the welcome message. Just type the same URL you set as the \$ baseURL. Once you do that, you should be directed to the welcome page of CodeIgniter. This should display the following message page-

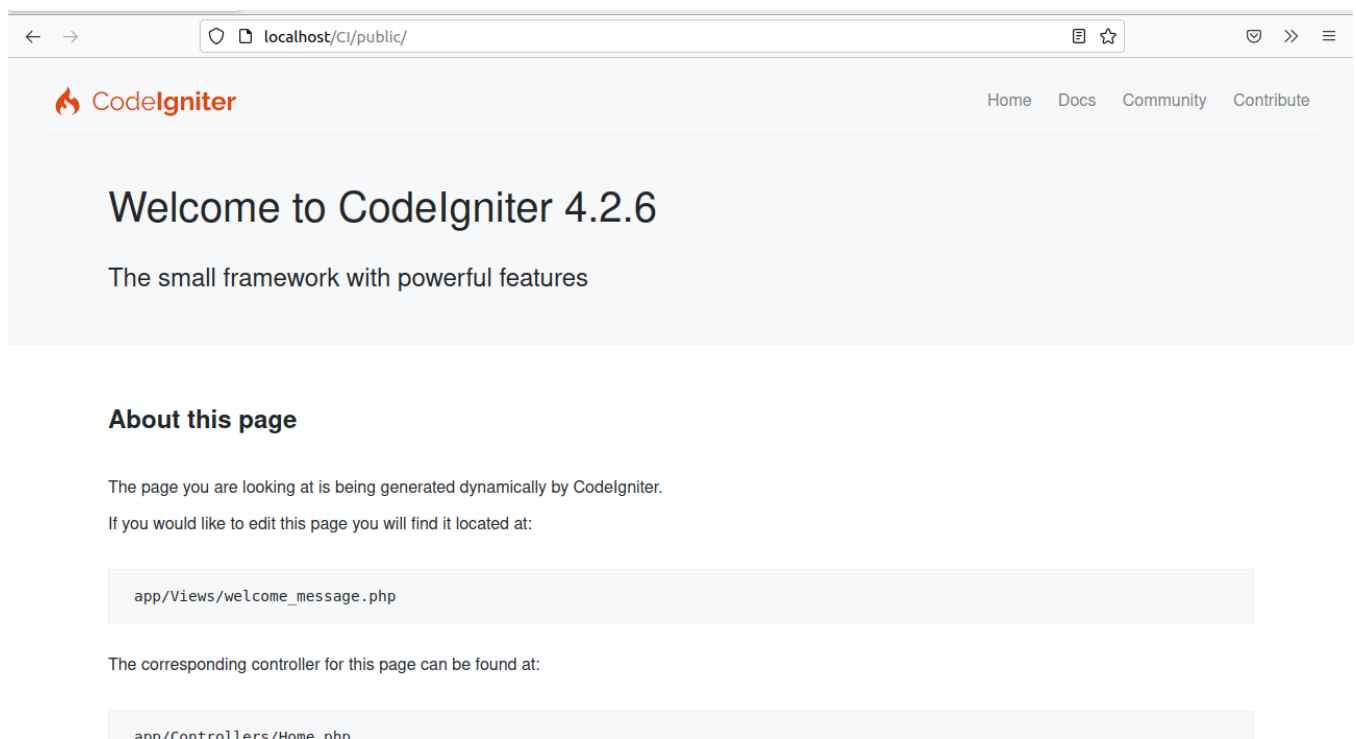


Fig 6.5: Displaying the welcome page

Practical No.7

To construct a simple login page web application to authenticate users using codeigniter framework and also perform CRUD operations.

To create a login system using codeigniter4, we follow the given steps-

- Open phpmyadmin and generate a new database by the name "ci4login". Add a new table to it by the name 'user'. Run the following SQL Query:

```
CREATE TABLE users (  
id INT PRIMARY KEY AUTO INCREMENT,  
name VARCHAR(150),  
email VARCHAR(150),  
password VARCHAR(150),  
created at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
) ENGINE=INNODB;
```

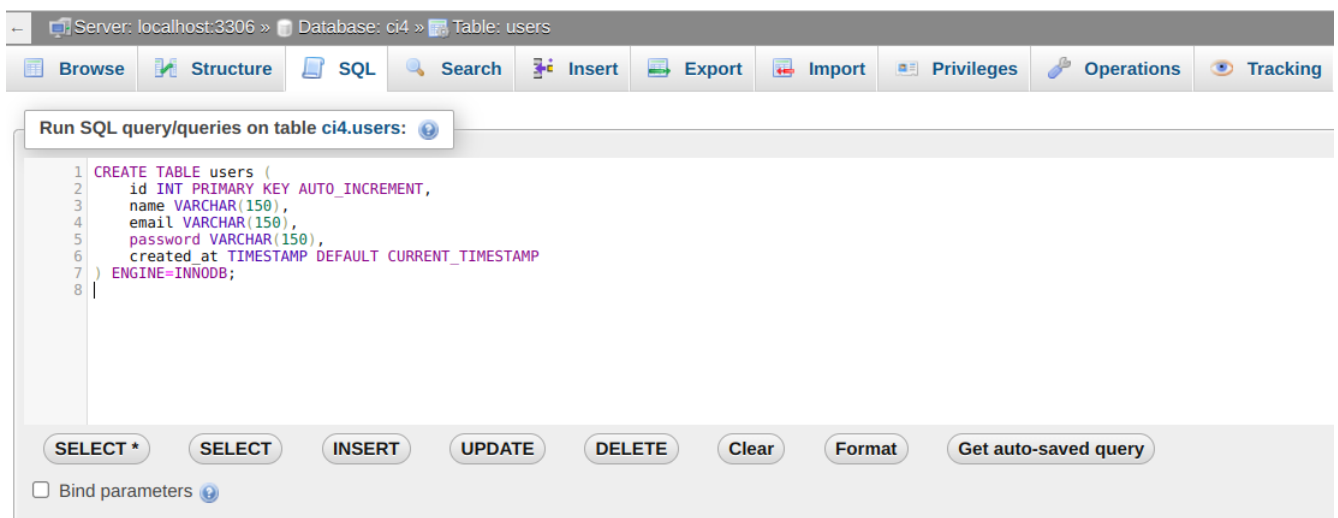


Fig 7.1 :Creating a table in SQL

- Open the app/Config/Database.php, and insert database name, username and password into the file.

```

*/
public $default = [
    'DSN' => '',
    'hostname' => 'localhost',
    'username' => 'root',
    'password' => '',
    'database' => 'ci4',
    'DBDriver' => 'MySQLi',
    'DBPrefix' => '',
    'pConnect' => false,
    'DBDebug' => (ENVIRONMENT !== 'development'),
    'charset' => 'utf8',
    'DBCollat' => 'utf8_general_ci',
    'swapPre' => '',
    'encrypt' => false,
    'compress' => false,
    'strictOn' => false,
    'failover' => [],
    'port' => 3306,
];

```

Fig 7.2 :Configuring the database

- Create the corresponding Model, Controllers and view files in the respective folders.
- In the next step, get inside the app/Config/Filters.php, look for aliases array and replace the whole array , protecting it with filter.

```

*/
public $aliases = [
    'csrf' => \CodeIgniter\Filters\CSRF::class,
    'toolbar' => \CodeIgniter\Filters\DebugToolbar::class,
    'honeypot' => \CodeIgniter\Filters\Honeypot::class,
    'authGuard' => \App\Filters\AuthGuard::class,
];

```

Fig 7.3 :Protecting the data with filter

- Define the routes for the created views in app/config/routes.php folder.

```

// $routes->get('/', 'news::index');
$routes->get('/', 'Home::index');
// $routes->get('/', 'frontPage::index');
// $routes->match(['get', 'post'], '/signinpage', 'signinpage::index');
// $routes->match(['get', 'post'], '/SignUpPage', 'SignUpPage::index');

$routes->get('/signup', 'SignupController::index');

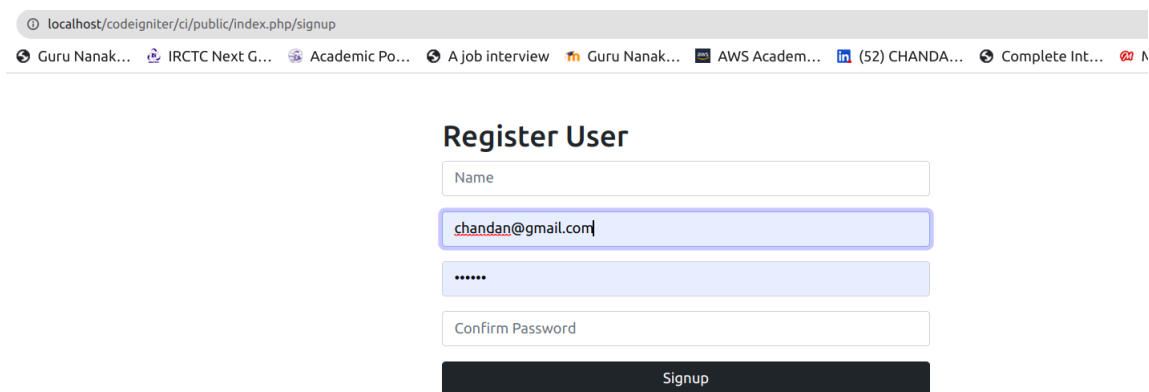
$routes->match(['get', 'post'], '/signup', 'SignupController::store');

$routes->get('/signin', 'SignInController::index');
$routes->match(['get', 'post'], '/signin', 'SignInController::loginAuth');
$routes->get('/profile', 'ProfileController::index', ['filter' => 'authGuard']);
$routes->get('/pages', 'Pages::index');
$routes->get('/:any', 'Pages::view/$1');
// $routes->get('/signup', 'SignupController::store');

```

Fig 7.4 :Defining the routes

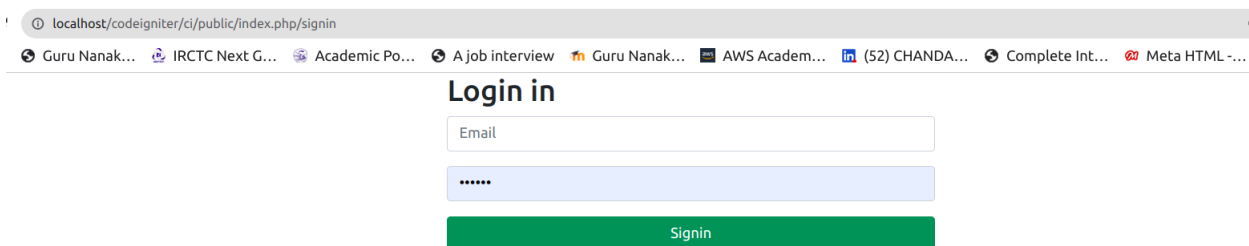
- Now we can go to the browser and type the URL and we should get the signup page.



The screenshot shows a web browser window with the address bar displaying 'localhost/codeigniter/ci/public/index.php/signup'. The browser's tab bar shows several open tabs, including 'Guru Nanak...', 'IRCTC Next G...', 'Academic Po...', 'A job interview', 'Guru Nanak...', 'AWS Academ...', '(52) CHANDA...', 'Complete Int...', and 'Meta HTML...'. The main content area displays a 'Register User' form. The form consists of four input fields: 'Name', 'Email' (containing 'chandan@gmail.com'), a password field (masked with dots), and a 'Confirm Password' field. Below these fields is a dark grey button labeled 'Signup'.

Fig 7.5 :Displaying the signup page.

- Once you signup, the data should go to the database and on signing up with the same information, we should be able to signin.



The screenshot shows a web browser window with the address bar displaying 'localhost/codeigniter/ci/public/index.php/signin'. The browser's tab bar shows several open tabs, including 'Guru Nanak...', 'IRCTC Next G...', 'Academic Po...', 'A job interview', 'Guru Nanak...', 'AWS Academ...', '(52) CHANDA...', 'Complete Int...', and 'Meta HTML...'. The main content area displays a 'Login in' form. The form consists of two input fields: 'Email' and a password field (masked with dots). Below these fields is a green button labeled 'Signin'.

Fig 7.6 : The signin page.

- After signing in, you should be directed to the welcome message.

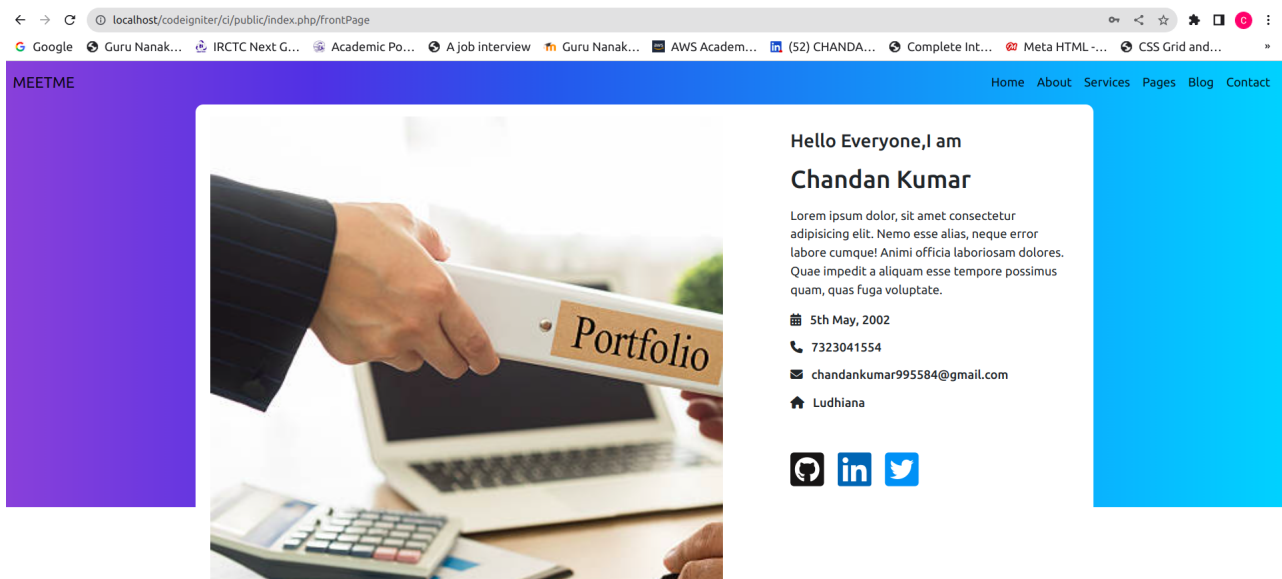


Fig 7.7: The welcome message.

Practical No.8

To install and setup,configure the laravel framework.

Laravel is an open-source PHP framework, which is robust and easy to understand. It follows a model-view-controller design pattern. Laravel reuses the existing components of different frameworks which helps in creating a web application. The web application thus designed is more structured and pragmatic.

Laravel offers a rich set of functionalities which incorporates the basic features of PHP frameworks like CodeIgniter, Yii and other programming languages like Ruby on Rails. Laravel has a very rich set of features which will boost the speed of web development.

Installation and Setup:

A) Installing Composer:

- Go to <https://getcomposer.org/> and select the Download option.

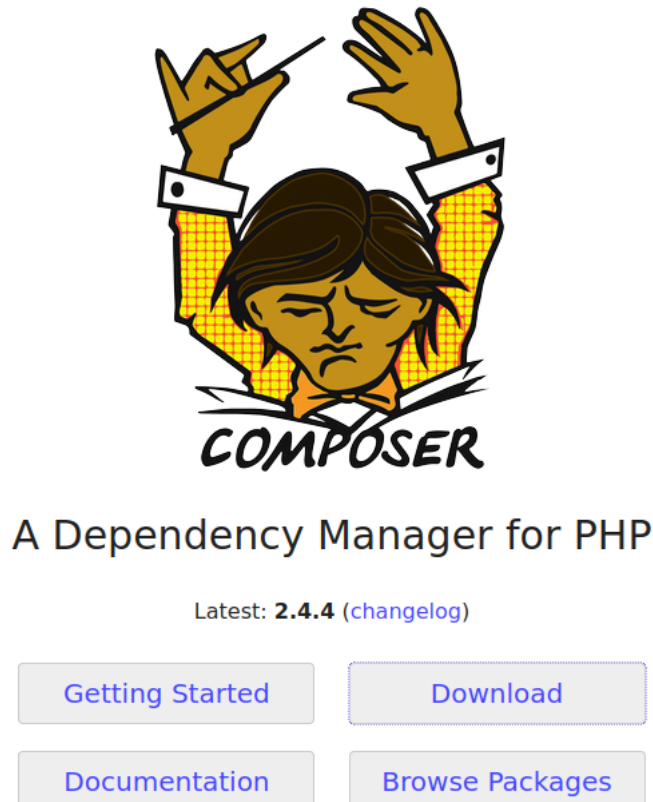


Fig 8.1: Downloading the Composer.

- Run the following given script in your terminal. This installer script will simply check some php.ini settings, warn you if they are set incorrectly, and then download the latest composer.phar in the current directory.

```
ubuntu@ubuntu1:/var/www/html/Laravel$ php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') === '55ce33d7678c5a611085589f1f3dddf8b3c52d662cd01d4ba75c0ee0459970c2200a51f492d557530c71
c15d8dba01eae') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
php composer-setup.php
php -r "unlink('composer-setup.php');"

Installer verified
All settings correct for using Composer
Downloading...

Composer (version 2.4.4) successfully installed to: /var/www/html/Laravel/composer.phar
Use it: php composer.phar

ubuntu@ubuntu1:/var/www/html/Laravel$
```

Fig 8.2: Running Script for Installation

- Run the following command on your terminal :
sudo mv composer.phar /usr/local/bin/composer

With this you can simply call composer from any directory (Global install).

With this we are done with the composer installation. Now we can setup our new project in Laravel.

B) Setting up a Laravel Project:

- Head over to <https://laravel.com/docs/9.x> and click on the Documentation button.

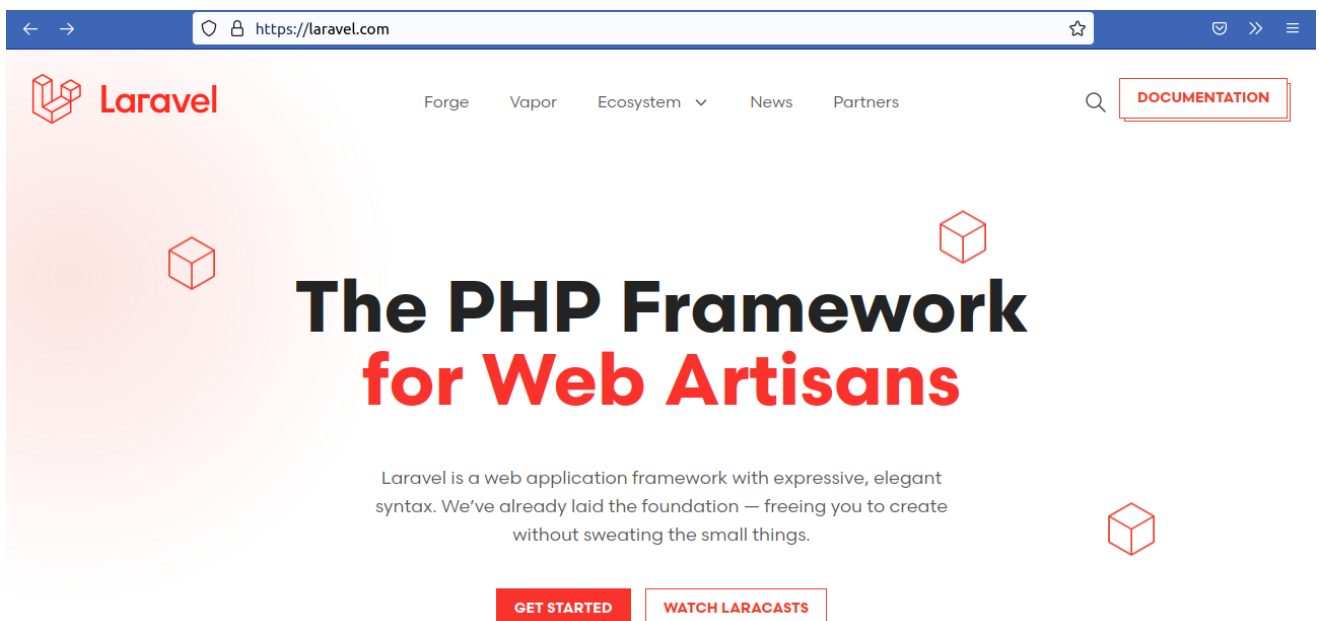
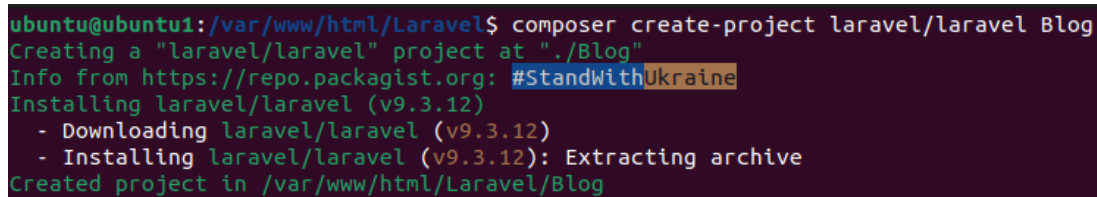


Fig 8.3: The official Laravel website

- Run the following command to create a Laravel Project :

composer create-project laravel/laravel Blog



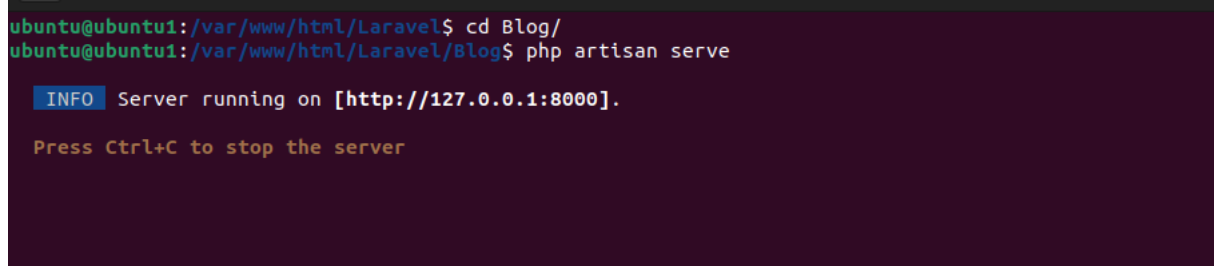
```
ubuntu@ubuntu1:/var/www/html/Laravel$ composer create-project laravel/laravel Blog
Creating a "laravel/laravel" project at "./Blog"
Info from https://repo.packagist.org: #StandWithUkraine
Installing laravel/laravel (v9.3.12)
- Downloading laravel/laravel (v9.3.12)
- Installing laravel/laravel (v9.3.12): Extracting archive
Created project in /var/www/html/Laravel/Blog
```

Fig 8.4: Creating a new Laravel Project

- Change Directory to the newly created Laravel Project . After this run the local server on your terminal:

cd Blog

php artisan serve



```
ubuntu@ubuntu1:/var/www/html/Laravel$ cd Blog/
ubuntu@ubuntu1:/var/www/html/Laravel/Blog$ php artisan serve

INFO Server running on [http://127.0.0.1:8000].

Press Ctrl+C to stop the server
```

Fig 8.5: Running the project on the local server.

- With this , you can visit the URL by entering it into your browser.This should display the welcome page of the Laravel Framework.

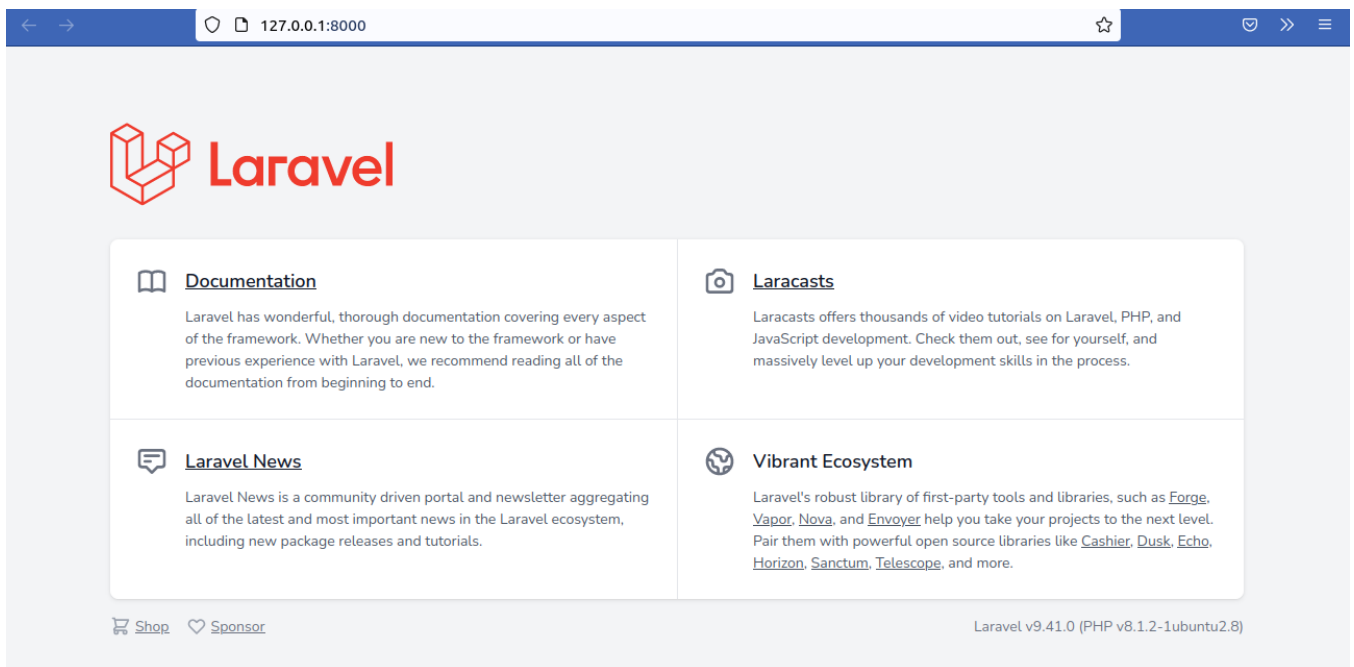


Fig 8.6: The welcome page running at the local server

Practical No.9

To construct any simple web application using Laravel Framework.

Creating a basic task list: Here we shall be creating a basic list that displays a list of tasks and allows users to add and delete tasks as per their need.

Once we are done with the basic setup and configurations , we need to follow the given steps -

- Configuring the Database : First, let's use a migration to define a database table to hold all of our tasks. Laravel's database migrations provide an easy way to define your database table structure and modifications using fluent, expressive PHP code.

Here we use the make:migration command to generate a new database migration for our tasks table. Type the following command in the terminal-

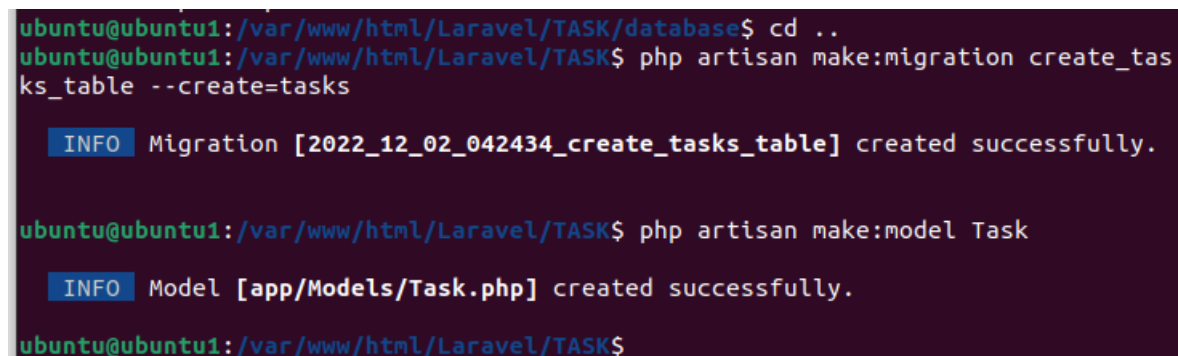
```
php artisan make:migration create taskstable --create=tasks
```

After the table is created, we need to use migrate command to be able to access it:

```
php artisan migrate
```

- Making the Model : Eloquent is Laravel's default ORM (object-relational mapper). Eloquent makes it painless to retrieve and store data in your database using clearly defined "models". Usually, each Eloquent model corresponds directly with a single database table. We use the following command to create a model

```
php artisan make:model Task
```



```
ubuntu@ubuntu1:/var/www/html/Laravel/TASK/database$ cd ..
ubuntu@ubuntu1:/var/www/html/Laravel/TASK$ php artisan make:migration create_tasks_table --create=tasks

  INFO  Migration [2022_12_02_042434_create_tasks_table] created successfully.

ubuntu@ubuntu1:/var/www/html/Laravel/TASK$ php artisan make:model Task

  INFO  Model [app/Models/Task.php] created successfully.

ubuntu@ubuntu1:/var/www/html/Laravel/TASK$
```

Fig 9: Making table and model

- Defining the controller : Laravel resource controllers provide the CRUD routes to the controller in a single line of code. A resource controller is used to create a controller that handles all the http requests stored by your application. We can create a resourceful controller by typing the following command in

- Defining the views : This application only has a single view which contains a form for adding new tasks as well as a listing of all current tasks. To help you visualize the view, here is a screenshot of the finished application with basic Bootstrap CSS styling applied.

We create three views for listing, editing and deleting the tasks. Each of the views should have the extension name.blade.php

- Defining the routes : Routes are used to point URLs to controllers or anonymous functions that should be executed when a user accesses a given page. By default, all Laravel routes are defined in the app/Http/routes.php file that is included in every new project.



```

15
16 Route::get('/', function () {
17     $tasks = Task::orderBy('created_at', 'asc')->get();
18
19     return view('tasks', [
20         'tasks' => $tasks
21     ]);
22 });
23
24 Route::post('/task', function (Request $request) {
25     $validator = Validator::make($request->all(), [
26         'name' => 'required|max:255',
27     ]);
28
29     if ($validator->fails()) {
30         return redirect('/')
31             ->withInput()
32             ->withErrors($validator);
33     }
34
35 Route::delete('/task/{id}', function ($id) {
36     Task::findOrFail($id)->delete();
37
38     return redirect('/');
39 });
40
41

```

Fig 9.2: Defining the routes

With this , we are set to run our application. To run using the local server type,
php artisan serve

Task List

New Task

Task

+ Add Task

Current Tasks

Task

First Task

Delete

Second Task

Delete

Fig 9.3: The task list

30