ADVANCED WEB TECHNOLOGIES LABORATORY LPEIT-102 PRACTICAL FILE

SUBMITTED TO:

DR. AKSHAY GIRDHAR

PROFESSOR (IT)

GNDEC, LUDHIANA

SUBMITTED BY:

SHUBHAM KUMAR

UNIVERSITY ROLL NO: 2104574

CLASS: IT (D3)

SECTION: B2

INDEX

S.NO	PRACTICAL	DATE	SIGN
1	To install and setup the HTML5 based Bootstrap framework and to deploy basic HTML elements using Bootstrap CSS.		
2	To understand and deploy the multicolumn grid layout of Bootstrap.		
3	To deploy different types of buttons, progress bars, modals and navigation bars using Bootstrap.		
4	To create and setup the Git repository on Bitbucket or Github using SSH.		
5	To perform push, c lone and patch operation to Git repository		
6	To install and setup the CodeIgniter Framework and to understand its MVC architecture.		
7	To construct a simple login page web application to authenticate users using CodeIgniter Framework and also perform CRUD operations.		
8	To install and setup, configure the Laravel Framework.		
9	To construct the any simple web application using Laravel Framework		

PRACTICAL 1: To install and setup the HTML5 based Bootstrap framework and to deploy basic HTML elements using Bootstrap CSS.

Bootstrap

Bootstrap is not just a statistical technique; it's also a popular front-end web development framework. Bootstrap, in the context of web development, is an open-source CSS (Cascading Style Sheets) framework that provides a set of pre-designed and reusable HTML and CSS components for building responsive and visually appealing websites and web applications.

Key Features of Bootstrap

- **Responsive Design:** Bootstrap is known for its responsive grid system, which allows developers to create web layouts that adapt seamlessly to various screen sizes and devices, such as desktops, tablets, and smartphones.

- **Typography:** Bootstrap offers a set of typography styles and classes for text formatting, making it easy to create consistent and visually pleasing text content.

- **CSS Components:** It includes a wide range of CSS components, including buttons, forms, navigation bars, alerts, badges, and more, all styled and designed to look cohesive out of the box.

- **JavaScript Plugins:** Bootstrap also includes JavaScript plugins that enhance the functionality of web elements. Examples include modal dialogs, carousels, tooltips, and popovers.

- **Customization:** Bootstrap can be customized to match the specific design requirements of a project. You can modify the framework's variables and styles to create a unique look and feel.

How To Setup Bootstrap?

There are two ways to install Bootstrap
- Link Bootstrap from a CDN
- Download Bootstrap.

Bootstrap CDN

Use Bootstrap CDN to quickly add Bootstrap to your project. It is as easy as just adding a few links to your web application and you would have successfully set up Bootstrap in your project.

Let's go through an example to create a simple Bootstrap-powered web page using Bootstrap CDN.

Step 1: creating a basic HTML file

Open up your code editor and create a new HTML file. Save the following code as "example.html" on your computer.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap Example</title>
</head>
<body>
    <h1>Let's build our first Bootstrap powered webpage using Bootstrap CDN!</h1>
</body>
</html>
```

Step 2: add Bootstrap CDN

Bootstrap CSS: Copy-paste the stylesheet <link> into your <head> before all other stylesheets to load Bootstrap CSS.

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
```

After adding the CDN links, the code looks like this. Save this new file as bootstrap_example.html.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap Example</title>
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
<body>
    <h1>Let's build our first Bootstrap powered webpage!</h1>
    <!-- Bootstrap JS Bundle -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>
</body>
</html>
```

Step 3: add Bootstrap classes

Let's use the Bootstrap Alert CSS class to use colors to add meaning to a text such as successor danger messageoralert`.

In the below code, the class " alert alert-success" is a pre-existing CSS class in Bootstrap that can be readily used by any user to create a success message or alert.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
```

```html
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Bootstrap Example</title>
  <!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
<body>
  <h1>Let's use Bootstrap Alert Component</h1>
  <div class="alert alert-primary" role="alert">
 A simple primary alert—check it out!
  </div>
  <div class="alert alert-secondary" role="alert">
    A simple secondary alert—check it out!
  </div>
  <div class="alert alert-success" role="alert">
    A simple success alert—check it out!
  </div>
  <div class="alert alert-danger" role="alert">
   A simple danger alert—check it out!
  </div>
  <div class="alert alert-warning" role="alert">
    A simple warning alert—check it out!
  </div>
  <div class="alert alert-info" role="alert">
    A simple info alert—check it out!
  </div>
  <div class="alert alert-light" role="alert">
    A simple light alert—check it out!
  </div>
  <div class="alert alert-dark" role="alert">
    A simple dark alert—check it out!
  </div>
  <!-- Bootstrap JS Bundle with Popper -->
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>
</body>
</html>
```
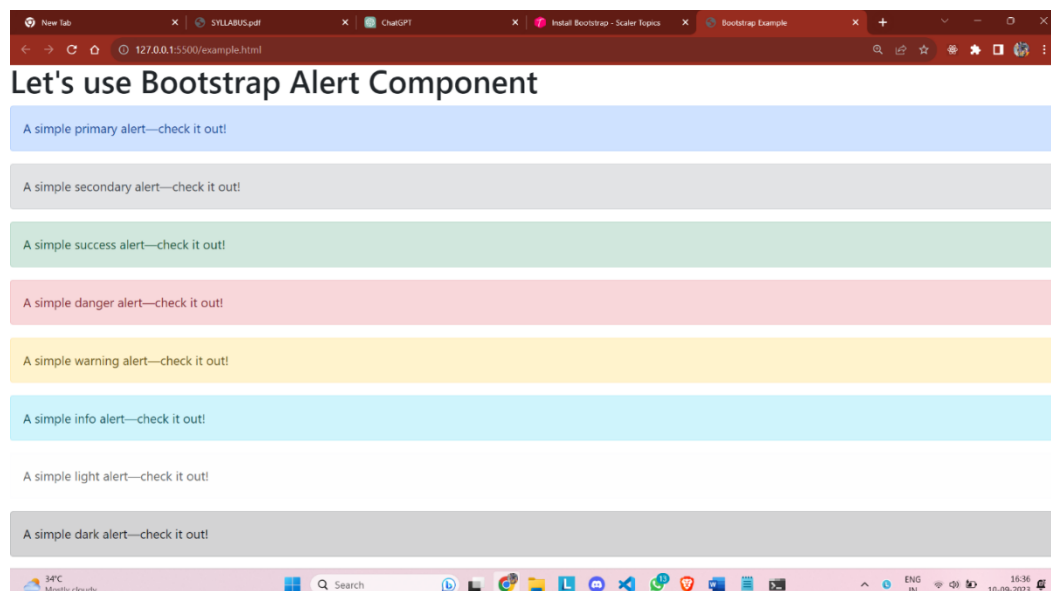

Congratulations! You have successfully set up Bootstrap in your web application using CDN.

Now, let's look at how to set up Bootstrap by downloading it.

Download Bootstrap

You can download Bootstrap's CSS and JavaScript files from their official website on your system and include them in your web application.

Let's go through the steps required to install Bootstrap.

Step 1: Download Bootstrap

To download the latest version of Bootstrap follow link given here Bootstrap Download Link. In the Download section, go to the Compiled CSS and JS section and click Download.

Compiled CSS and JS

Download ready-to-use compiled code for **Bootstrap v5.3.1** to easily drop into your project, which includes:

- Compiled and minified CSS bundles (see CSS files comparison)
- Compiled and minified JavaScript plugins (see JS files comparison)

This doesn't include documentation, source files, or any optional JavaScript dependencies like Popper.

Download

Once the file has been downloaded, go to the file location on your device and unzip the files.

The contents include:

- Compiled CSS and JS (bootstrap.*) files
- Compiled and minified CSS and JS (bootstrap.min.*`)
- Source maps (bootstrap.*.map) are available for use with certain browser developer tools.
- Bundled JS files (bootstrap.bundle.js and minified bootstrap.bundle.min.js) which include Popper.

Based on the requirements and resources of the web application, the file to be used can be chosen.

Let's use the compiled and minified CSS and JS files for the example.

Step 2: copy contents in the same directory as HTML document

Create "example.html" file on your computer. Copy the contents of the unzipped files i.e CSS and js folders in the same directory as "example.html".

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Bootstrap Example</title>
</head>
<body>
  <h1>Let's build our first Bootstrap powered webpage using Bootstrap Download!</h1>
</body>
</html>
```

Step 3: load Bootstrap CSS and JS file in the HTML document

To load the Bootstrap CSS, go to the CSS folder that you copied and find the file "bootstrap.min.css". Now, copy the path of the file and paste it into the href section below.

```
<link rel="stylesheet" href="bootstrap/css/bootstrap.min.css">
```

Similarly to load Bootstrap JS, go to the js folder and copy the path of the bootstrap.min.js file and paste it into the src section of the below code.

```
<script src="bootstrap/js/bootstrap.min.js"></script>
```

Place the following <script> near the end of your example.html file, right before the closing </body> tag.

Once you put it all together, the code looks like this:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Bootstrap Example</title>
  <!-- Bootstrap CSS -->
  <link rel="stylesheet" href="bootstrap/css/bootstrap.min.css">
</head>
<body>
  <h1>Let's build our first Bootstrap powered webpage using Bootstrap Download!</h1>
<!-- Bootstrap JS -->
<script src="bootstrap/js/bootstrap.min.js"></script>
</body>
</html>
```


New in v5.3 Color mode support, expanded color palette, and more!

B

Build fast, responsive sites with Bootstrap

Powerful, extensible, and feature-packed frontend toolkit. Build and customize with Sass, utilize prebuilt grid system and components, and bring projects to life with powerful JavaScript plugins.

$ npm i bootstrap@5.3.1   Read the docs
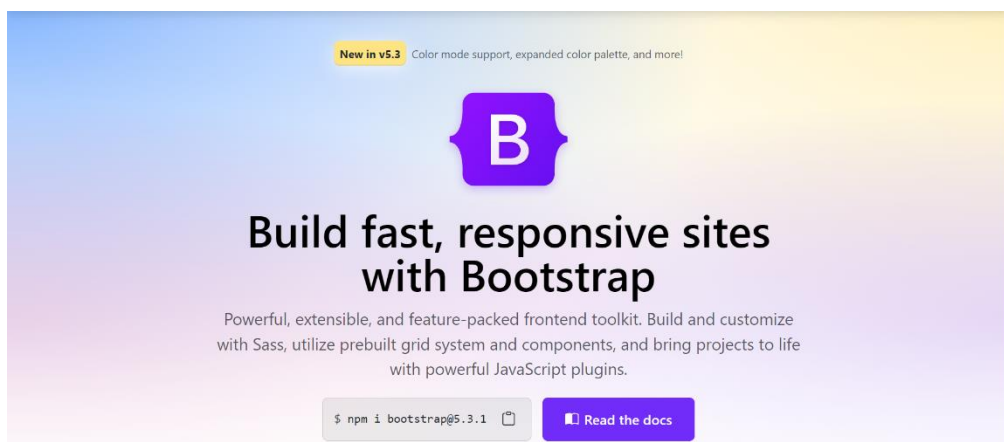
PRACTICAL 2: To understand and deploy the multicolumn grid layout of Bootstrap.

Multicolumn Grid Layout Key Features

- **Responsive Grid System:** Bootstrap provides a responsive grid system that helps create multicolumn layouts that adapt to various screen sizes.

- **Container:** Use the .container class to create a container for your grid. It provides a fixed-width container centered on the page.

- **Row:** Within the container, use the .row class to create a row for your columns. Rows are used to group columns horizontally.

- **Columns:** Inside a row, use .col-* classes to define the columns. The * represents the number of columns you want in each row, and Bootstrap supports a 12-column grid system.

- **Responsive Classes:** You can use responsive classes like .col-md-* to control column behavior at different screen sizes. In this example, md stands for medium-sized screens, and you can adjust it to sm, lg, or xl for different breakpoints.

- **Nested Grids:** You can also create nested grids by placing rows and columns inside other columns to create complex layouts.

- **Content:** Add your content, such as text, images, or other HTML elements, within the columns.

- **Bootstrap JavaScript:** Including Bootstrap JavaScript is optional and is mainly used for certain components and interactions. It's added using <script> tags.

- **Testing:** Test your layout by resizing the browser window to see how it responds to different screen sizes.

- **Custom Styling:** Customize the appearance of your columns and content using your own CSS styles in an external stylesheet.

Multiple Column Layouts using Bootstrap

Up	Cars	Rio
Up is a 2009 American computer-animated comedy-drama adventure film.	Cars is a 2006 American computer-animated comedy film.	Rio is a 2011 American 3D computer-animated musical adventure comedy film.

Available breakpoints

Bootstrap includes six default breakpoints, sometimes referred to as *grid tiers*, for building responsively. These breakpoints can be customized if you're using our source Sass files.

Breakpoint	Class infix	Dimensions
Extra small	None	<576px
Small	sm	≥576px
Medium	md	≥768px
Large	lg	≥992px
Extra large	xl	≥1200px
Extra extra large	xxl	≥1400px

Basic Structure of a Multicolumn Grid Layout

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Multi-Column Grid Layout</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
</head>
<body class="bg-dark">
  <div class="container bg-info">
    <h1>Multi-Column Grid Layout</h1>
    <div class="row">
      <div class="col-md-4">
        <div class="card">
          <img src="./images/col1.jpeg" alt="">
          <div class="card-body">
            <h5 class="card-title">Column 1</h5>
            <p class="card-text">Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
          </div>
        </div>
      </div>
      <div class="col-md-4">
        <div class="card">
          <img src="./images/col2.jpeg" class="card-img-top" alt="Image 2">
          <div class="card-body">
            <h5 class="card-title">Column 2</h5>
            <p class="card-text">Suspendisse nec felis a tellus consequat tincidunt.</p>
          </div>
        </div>
      </div>
      <div class="col-md-4">
        <div class="card">
          <img src="./images/col1.jpeg" class="card-img-top" alt="Image 3">
          <div class="card-body">
            <h5 class="card-title">Column 3</h5>
            <p class="card-text">Vestibulum auctor erat eu orci bibendum, eget eleifend velit dictum.</p>
          </div>
        </div>
      </div>
    </div>
    <br><br><br>
  </div>
</body>
</html>
```
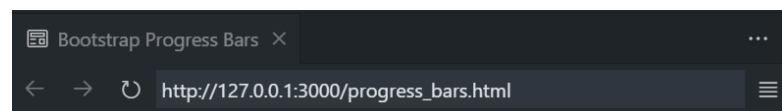
Output md screen(>=768px)   Output sm screen(<768px)

Practice 3: To deploy different types of buttons, progress bars, modals and navigation bars using Bootstrap.

Deployment of Bootstrap Components – Introduction

Bootstrap is a popular front-end framework that simplifies the process of creating responsive and visually appealing web applications. This practical exercise aims to demonstrate how to deploy various Bootstrap components, including buttons, progress bars, modals, and navigation bars, in a web project. These components play a crucial role in enhancing user experience and interactivity on web applications.

Materials Required

1. - A computer with a text editor and web browser
2. - Internet connection (for linking Bootstrap via CDN)

Procedure

Step 1: Setting up Bootstrap

1. Begin by creating an HTML document (e.g., `index.html`) using your preferred text editor.
2. Link the Bootstrap CSS and JavaScript files by adding the following code within the
```
 <head>` section of your HTML document:
  <!-- Link to Bootstrap CSS -->
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">

  <!-- Link to Bootstrap JS and Popper.js for some Bootstrap features -->
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.
js"></script>
```

Step 2: Deploying Buttons

1. Create buttons using Bootstrap classes. For example:

```
<button class="btn btn-primary">Primary Button</button>
<button class="btn btn-secondary">Secondary Button</button>
<button class="btn btn-success">Success Button</button>
```

2. Customize the buttons by experimenting with different classes and styles provided by Bootstrap.
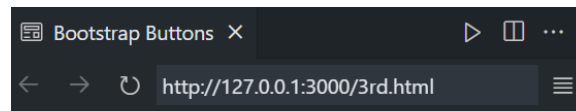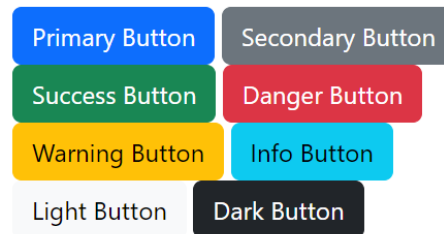
Step 3: Creating Progress Bars

1. **Develop a progress bar using Bootstrap's classes. For instance:**

html
```
  <div class="progress">
    <div class="progress-bar" style="width: 70%;" aria-valuenow="70" aria-valuemin="0" aria-valuemax="100"></div>
  </div>
```

2. Adjust the progress bar's width and appearance to suit your application's needs.
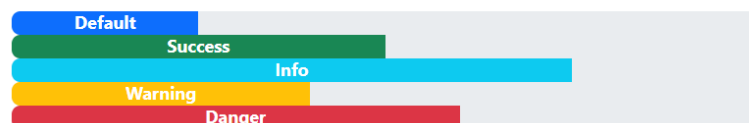
Step 4: Designing Modals

1. Create a modal dialog using Bootstrap. This modal can display additional information or perform specific actions when triggered. Example modal code:

```
  <!-- Button to trigger modal -->
  <button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-target="#myModal">
    Open Modal
  </button>
```

```
<!-- Modal -->
<div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-
labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog" role="document">
    <!-- Modal content goes here -->
  </div>
</div>
```

2. Customize the modal's content and appearance based on your project's requirements.
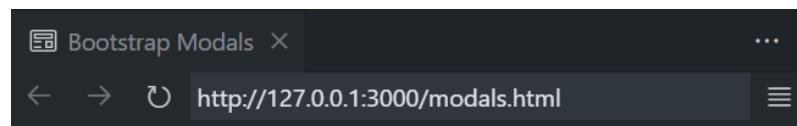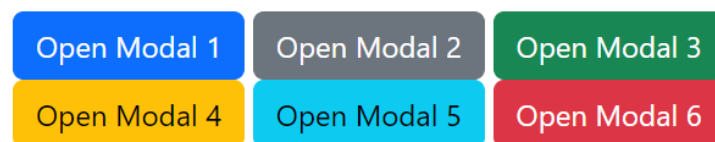
Step 5: Implementing Navigation Bars

1. Create a navigation bar using Bootstrap. A navigation bar is essential for providing easy access to various sections of your website. Example navigation bar code:

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <!-- Navbar content goes here -->
</nav>
```
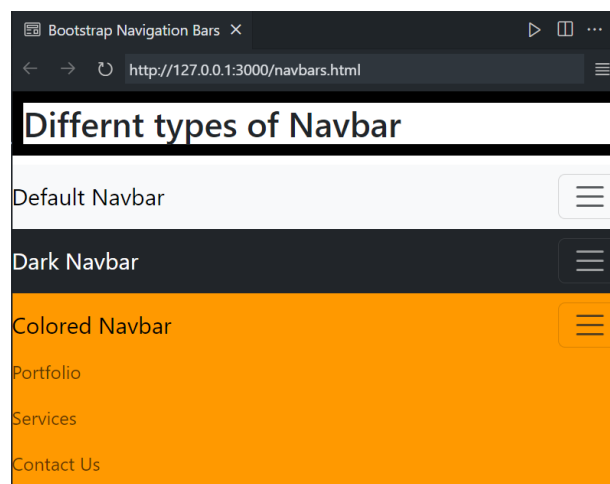
Practical 4: To create and setup the Git repository on Bitbucket or GitHub using SSH.

SSH (Secure Shell Protocol)

Using the SSH protocol, you can connect and authenticate to remote servers and services. With SSH keys, you can connect to GitHub without supplying your username and personal access token at each visit. You can also use an SSH key to sign commits.
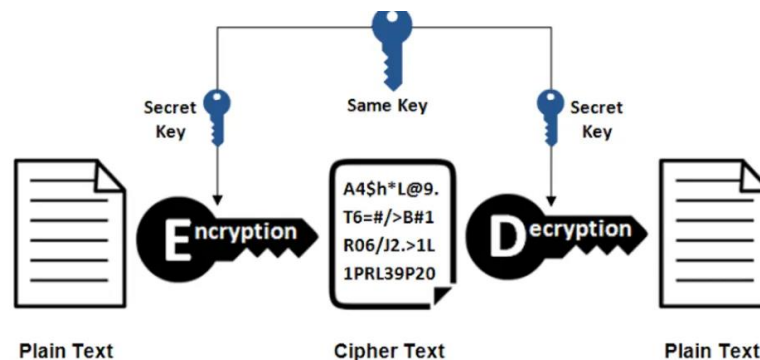
To create and set up a Git repository on GitHub using SSH, you'll need to follow these steps:

1. **Generate an SSH Key:** If you haven't already, you'll need to generate an SSH key pair. This consists of a public key (which you'll upload to GitHub) and a private key (which you'll keep on your local machine). You can generate an SSH key using the following command in your terminal:

   ```
   ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
   ```

2. **Add your SSH Key to the SSH Agent:** You'll want to add your private key to the SSH agent to manage your keys. You can do this with the following command

   ```
   eval "$(ssh-agent -s)"
   ssh-add ~/.ssh/id_rsa
   ```

3. **Copy your Public Key:** Use the following command to copy your public key to your clipboard:

   ```
   pbcopy < ~/.ssh/id_rsa.pub
   ```

4. **Add SSH Key to Your GitHub Account:** Log in to your GitHub account and go to "Settings" > "SSH and GPG keys." Click on the "New SSH key" button

and paste your public key into the provided field. Give it a descriptive title to help you identify it.

5. **Create a New Repository:** On GitHub, click on the "+" icon in the upper right corner and select "New repository." Fill in the repository name, description, and other details as needed.
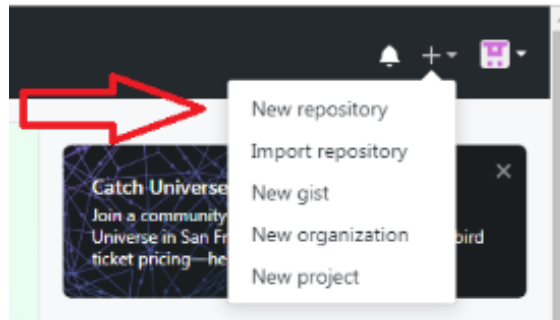
6. **Initialize Your Local Repository:** On your local machine, navigate to the directory where you want to create your Git repository. Run the following commands:

```
git init //initialise your repo

git add . //Stage Files

git commit -m "Initial commit" //Commit Files
```

7. **Set the Remote Origin:** On the GitHub repository page, you'll see a section called "Quick setup." Copy the URL provided for the repository, which should start with "git@github.com." In your local terminal, set the remote origin using the following command (replace the URL with your repository's URL):

```
git remote add origin git@github.com:yourusername/your-repo.git
```

8. **Push Your Code to GitHub:** Finally, push your code to the GitHub repository using the following command:

```
git push -u origin master
```

Practical 5: To perform push, clone and patch operation to Git repository.

PUSH

In Git, "push" is a command used to send your locally committed changes to a remote repository, typically hosted on a platform like GitHub, GitLab, Bitbucket, or a private Git server. When you push your changes, you're essentially updating the remote repository with the latest commits from your local repository. This is a fundamental part of collaborative development and version control in Git. Here's how the git push command works:

1. **Local Repository:** You have a Git repository on your local machine where you work on your code.
2. **Remote Repository:** This is a Git repository hosted on a server, which could be a platform like GitHub, GitLab, or a company's internal Git server. Remote repositories allow multiple developers to collaborate on a project.
3. **git commit:** Before you can push your changes, you need to commit them to your local repository using the git commit command. This creates a new commit with the changes you've made.
4. **git push:** Once you have local commits that you want to share with others or store in a remote repository, you use the git push command. The basic syntax for pushing is:

```
git push <remote> <branch>
```

- **<remote>** is the name of the remote repository (often named "origin" by default), which represents the URL of the remote server where your code is hosted.
- **<branch>** is the branch you want to push. It's often "master" for the main branch, but you can push any branch.

For example, if you want to push your local "master" branch to the "origin" remote repository:

```
git push origin master
```

When you push, Git uploads your commits to the remote repository, making your changes available to others who can then pull those changes into their own local repositories. This process is a central part of collaboration and keeping all contributors in sync.

Clone

In Git, "clone" is a command used to create a copy of a remote Git repository on your local machine. This command is commonly used when you want to start working on an existing project hosted on a remote Git server (such as GitHub, GitLab, Bitbucket, or a private Git server). Cloning a repository allows you to have a local copy of the

codebase, complete with the commit history, branches, and all the version control features provided by Git. Here's how you use the git clone command:

```
git clone <repository URL> <optional directory>
```

- **<repository URL>** is the URL of the remote Git repository you want to clone. It can be an HTTPS or SSH URL, depending on your authentication and access settings.
- **<optional directory>** is the name of the local directory where you want to clone the repository. If you don't specify a directory name, Git will create a new directory with the same name as the remote repository.
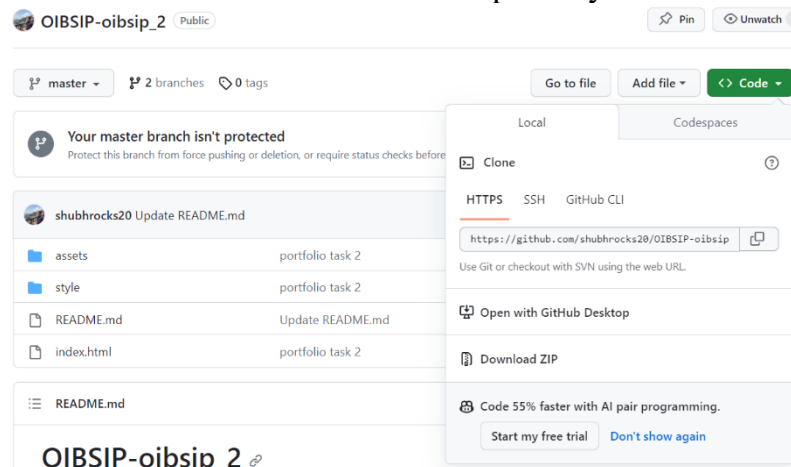
Patch

Git patch is a feature in git which enables you to create a patch file from a feature in one branch and apply it in another branch.
A patch file has all the differences between the two branches. Using the patch file, we can apply the changes in a different branch.

1. Creating a Patch:
   a. Make changes to your code.
   b. Stage the changes with git add.
   c. Create a patch using git format-patch:

```
git format-patch -1  # -1 indicates the last commit
```

This will create a patch file in the current directory. If you want to create a patch for a specific commit, replace -1 with the commit hash.

2. Applying a Patch:
   a. Ensure you have the patch file you want to apply.
   b. Use the git apply command:

```
git apply patchfile.patch
```

Practical 6: To Install and setup the CodeIgniter Framework and to understand its MVC architecture.

Download CodeIgniter:

- Visit the [CodeIgniter website](#) and download the latest version.

CodeIgniter 4 is the latest version of the framework, intended for use with PHP 7.4+ (including 8.2).

The initial release was February 24, 2020. The current version is v4.4.1.

You "could" download this version of the framework using the button below, but we encourage you to check the Installation section of the User Guide, to use Composer :)

CODEIGNITER 4

Download Discuss Sources Translations User Guide

Extract Files:

- Once downloaded, extract the files to your desired directory.

📁 codeigniter4-framework-v4.4.1-0-g84461...	15-10-2023 09:01	File folder	
📄 ADVANCED WEB TECH LAB	15-10-2023 22:37	Microsoft Word D...	1,743 KB
🗜️ codeigniter4-framework-v4.4.1-0-g84461...	15-10-2023 09:00	WinRAR ZIP archive	1,111 KB

Configure Base URL:

```
public string $baseURL = 'http://localhost:8080/codeigniter/';
```

Model:

1. **Responsibility**:
   - The Model is responsible for managing the data, its structure, and the operations that can be performed on it. This includes retrieving data from a database, performing calculations, and applying business logic.
2. **Characteristics**:
   - **Data Representation**: It represents the data in the application. This could be data retrieved from a database, an API, or any other source.

- **Data Operations**: It handles operations like reading, writing, updating, and deleting data. It ensures data integrity and consistency.
- **Business Logic**: Contains the application's business logic, which is the set of rules that govern how data is processed and manipulated.
- **No Knowledge of UI**: The Model has no knowledge of how data is presented or displayed to the user.
- **Observable**: In some implementations, Models can notify interested parties (observers) when their state changes.

View:

1. **Responsibility**:
   - The View is responsible for presenting data to the user in a human-readable format. It displays the user interface, including buttons, forms, text, and graphics.
2. **Characteristics**:
   - **User Interface**: It encompasses all the elements that the user interacts with, such as buttons, forms, text fields, and graphics.
   - **Presentation Logic**: Contains code for rendering data in a specific way. This could involve formatting dates, currency, or other data for display.
   - **No Business Logic**: The View should not contain any business logic. It's purely concerned with displaying data.
   - **No Direct Interaction with Data**: It does not directly interact with the database or manipulate data. It receives data from the Controller.
   - **Observable (Optional)**: In some implementations, Views can be observers that are notified when the underlying data changes.

Controller:

1. **Responsibility**:
   - The Controller acts as an intermediary between the Model and the View. It receives input from the user, processes that input (potentially interacting with the Model), and determines what the View should display.
2. **Characteristics**:
   - **Input Handling**: Listens for user input, which could be a mouse click, keyboard event, or any form of interaction with the application.

- **Business Logic Orchestration**: Orchestrates the flow of data between the Model and the View, applying business logic when needed.
- **Determines the View**: Decides which View should be displayed based on the user's actions or the state of the application.
- **No UI Logic**: While it may decide which View to display, it doesn't contain UI code. It delegates that responsibility to the View.
- **Updates the Model**: In response to user actions, the Controller may instruct the Model to update its state.

Flow of Data:

1. **User Interaction**:
   - The user interacts with the View, such as clicking a button or filling out a form.
2. **Controller Receives Input**:
   - The Controller receives the input and decides how to handle it.
3. **Controller Updates Model (Optional)**:
   - Depending on the action, the Controller may instruct the Model to update its data.
4. **Controller Determines View**:
   - The Controller decides which View to display based on the user's action and the state of the application.
5. **View Displays Data**:
   - The View receives data from the Controller and renders it for the user.
6. **User Sees Updated Interface**:
   - The user sees the updated interface in the View.
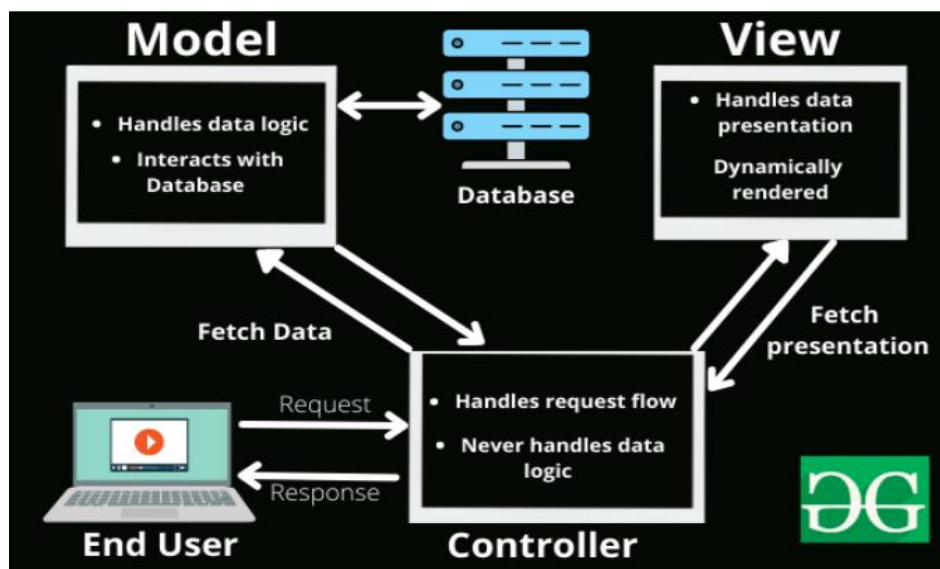
Benefits of MVC:

- **Separation of Concerns (SoC)**: It separates the application logic into distinct parts, making it easier to manage and maintain.
- **Code Reusability**: Components can be reused across different parts of the application or in different applications.
- **Testability**: Each component (Model, View, Controller) can be tested independently, which makes it easier to identify and fix issues.
- **Scalability**: It allows for scaling different components independently. For example, if the application needs to handle more traffic, you can scale the Controller or the View layer independently of the Model.
- **Flexibility**: Changes in one component do not necessarily affect the others, providing flexibility in development and maintenance.
- **Collaboration**: Different teams or developers can work on different components simultaneously, as long as they adhere to the defined interfaces.

Practical 7: To construct a simple login page web application to authenticate users using CodeIgniter Framework and also perform CURD operations.

CURD stands for Create, Update, Read and Delete.

- Create a new project using composer.
- Start the Xampp server
- Create a database named as demo

We will create a form for user details in which users can add their details and can update, delete and view those details.

1.Create: This will take input from user in a form.

2.Update: This will enable user to update the details.

3.Read: This will display the details of users in a table.

4.Delete: This will enable user to delete their details.

Firstly create views:

1.add_user.php:

```
<!DOCTYPE html>

<html>

<head>

 <title>Codeigniter 4 Add User With Validation Demo</title>

 <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">

 <style>

  .container {

   max-width: 500px;

  }

  .error {

   display: block;

   padding-top: 5px;

   font-size: 14px;

   color: red;
```

```
    }
  </style>
</head>
<body>
  <div class="container mt-5">
    <form method="post" id="add_create" name="add_create"
    action="<?= site_url('/submit-form') ?>">
      <div class="form-group">
        <label>Name</label>
        <input type="text" name="name" class="form-control">
      </div>
      <div class="form-group">
        <label>Email</label>
        <input type="text" name="email" class="form-control">
      </div>
      <div class="form-group">
        <button type="submit" class="btn btn-primary btn-block">Update Data</button>
      </div>
    </form>
  </div>
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-validate/1.19.2/jquery.validate.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-validate/1.19.2/additional-methods.min.js"></script>
  <script>
    if ($("#add_create").length > 0) {
      $("#add_create").validate({
        rules: {
```

```
          name: {
            required: true,
          },
          email: {
            required: true,
            maxlength: 60,
            email: true,
          },
        },
        messages: {
          name: {
            required: "Name is required.",
          },
          email: {
            required: "Email is required.",
            email: "It does not seem to be a valid email.",
            maxlength: "The email should be or equal to 60 chars.",
          },
        },
      })
    }
  </script>
</body>
</html>
```

2. edit_view.php:

```
<!DOCTYPE html>
<html>
<head>
  <title>Codeigniter 4 CRUD - Edit User Demo</title>
```

```html
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">
  <style>
    .container {
      max-width: 500px;
    }
    .error {
      display: block;
      padding-top: 5px;
      font-size: 14px;
      color: red;
    }
  </style>
</head>
<body>
  <div class="container mt-5">
    <form method="post" id="update_user" name="update_user"
    action="<?= site_url('/update') ?>">
      <input type="hidden" name="id" id="id" value="<?php echo $user_obj['id']; ?>">
      <div class="form-group">
        <label>Name</label>
        <input type="text" name="name" class="form-control" value="<?php echo
$user_obj['name']; ?>">
      </div>
      <div class="form-group">
        <label>Email</label>
        <input type="email" name="email" class="form-control" value="<?php echo
$user_obj['email']; ?>">
      </div>
      <div class="form-group">
```

```html
      <button type="submit" class="btn btn-danger btn-block">Save Data</button>
    </div>
  </form>
</div>
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-validate/1.19.2/jquery.validate.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-validate/1.19.2/additional-methods.min.js"></script>
<script>
  if ($("#update_user").length > 0) {
    $("#update_user").validate({
      rules: {
        name: {
          required: true,
        },
        email: {
          required: true,
          maxlength: 60,
          email: true,
        },
      },
      messages: {
        name: {
          required: "Name is required.",
        },
        email: {
          required: "Email is required.",
          email: "It does not seem to be a valid email.",
```

```
              maxlength: "The email should be or equal to 60 chars.",
            },
          },
        })
      }
    </script>
  </body>
</html>
```

3.user_view.php:

```
                                                    <!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <title>Codeigniter 4 CRUD App Example - positronx.io</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">
  </head>
<body>
<div class="container mt-4">
    <div class="d-flex justify-content-end">
        <a href="<?php echo site_url('/user-form') ?>" class="btn btn-success mb-2">Add User</a>
                                                    </div>
    <?php
    if(isset($_SESSION['msg'])){
        echo $_SESSION['msg'];
```

```php
    }
  ?>
 <div class="mt-3">
   <table class="table table-bordered" id="users-list">
    <thead>
      <tr>
        <th>User Id</th>
        <th>Name</th>
        <th>Email</th>
        <th>Action</th>
      </tr>
    </thead>
    <tbody>
      <?php if($users): ?>
      <?php foreach($users as $user): ?>
      <tr>
        <td><?php echo $user['id']; ?></td>
        <td><?php echo $user['name']; ?></td>
        <td><?php echo $user['email']; ?></td>
        <td>
        <a href="<?php echo base_url('edit_view/'.$user['id']);?>" class="btn btn-primary btn-sm">Edit</a>
        <a href="<?php echo base_url('delete/'.$user['id']);?>" class="btn btn-danger btn-sm">Delete</a>
        <a href="<?php echo base_url('delete/'.$user['id']);?>" class="btn btn-danger btn-sm">Delete</a>
        </td>
      </tr>
      <?php endforeach; ?>
      <?php endif; ?>
```

```html
      </tbody>
    </table>
  </div>
</div>


<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<link rel="stylesheet" type="text/css"
href="https://cdn.datatables.net/1.10.21/css/jquery.dataTables.min.css">

<script type="text/javascript"
src="https://cdn.datatables.net/1.10.21/js/jquery.dataTables.min.js"></script>
<script>
   $(document).ready( function () {

     $('#users-list').DataTable();

  } );
</script>
</body>
</html>
```

Models:

UserModel.php:

```php
<?php
namespace App\Models;
use CodeIgniter\Model;
class UserModel extends Model
{
   protected $table = 'users';
   protected $primaryKey = 'id';


   protected $allowedFields = ['name', 'email'];
```

}

Controller:

UserCrud.php

```php
<?php
namespace App\Controllers;
use App\Models\UserModel;
use CodeIgniter\Controller;
class UserCrud extends Controller
{
    // show users list
    public function index(){
        $userModel = new UserModel();
        $data['users'] = $userModel->orderBy('id', 'DESC')->findAll();
        return view('user_view', $data);
    }
    // add user form
    public function create(){
        return view('add_user');
    }

    // insert data
    public function store() {
        $userModel = new UserModel();
        $data = [
            'name' => $this->request->getVar('name'),
            'email'  => $this->request->getVar('email'),
        ];
        $userModel->insert($data);
```

```php
        return $this->response->redirect(site_url('/users-list'));
    }
    // show single user
    public function singleUser($id = null){
        $userModel = new UserModel();
        $data['user_obj'] = $userModel->where('id', $id)->first();
        return view('edit_view', $data);
    }
    // update user data
    public function update(){
        $userModel = new UserModel();
        $id = $this->request->getVar('id');
        $data = [
            'name' => $this->request->getVar('name'),
            'email'  => $this->request->getVar('email'),
        ];
        $userModel->update($id, $data);
        return $this->response->redirect(site_url('/users-list'));
    }


    // delete user
    public function delete($id = null){
        $userModel = new UserModel();
        $data['user'] = $userModel->where('id', $id)->delete($id);
        return $this->response->redirect(site_url('/users-list'));
    }
}
```

Create the routes:

```php
<?php

use CodeIgniter\Router\RouteCollection;

/**
 * @var RouteCollection $routes
 */
// CRUD RESTful Routes
$routes->get('/', 'UserCrud::index');
$routes->get('users-list', 'UserCrud::index');
$routes->get('user-form', 'UserCrud::create');
$routes->post('submit-form', 'UserCrud::store');
$routes->get('edit_view/(:num)', 'UserCrud::singleUser/$1');
$routes->post('update', 'UserCrud::update');
$routes->get('delete/(:num)', 'UserCrud::delete/$1');
```

In database.php add the database:

```php
public array $default = [
    'DSN'       => '',
    'hostname'  => 'localhost',
    'username'  => 'root',
    'password'  => '',
    'database'  => 'demo',
    'DBDriver'  => 'MySQLi',
    'DBPrefix'  => '',
    'pConnect'  => false,
    'DBDebug'   => true,
```

```
'charset'      => 'utf8',
'DBCollat'     => 'utf8_general_ci',
'swapPre'      => '',
'encrypt'      => false,
'compress'     => false,
'strictOn'     => false,
'failover'     => [],
'port'         => 3306,
'numberNative' => false,
];
```

Add User:

User View:

Update data:

Add new user:

New user added:

Deleted the users:

Practical 8: To install and setup, configure the Laravel Framework.

Composer is a dependency manager for PHP. It helps PHP developers manage and install external libraries or packages into their projects. Composer simplifies the process of including external libraries in your PHP application by handling the installation, autoloading, and dependency resolution.

First of all we need to install Composer in our system.

If we are getting this type of interface then we are done with our work.

To create Laravel Project

composer create-project laravel/laravel myProject

This will create a Laravel project in our desired folder we are in.

cd myProject

After this we will move in our Project directory.

After this we need to run this command

php artisan serve

```
PS C:\Users\princ\documents\laravel\myProject> php artisan serve
Starting Laravel development server: http://127.0.0.1:8000
[Fri Dec 15 06:59:09 2023] PHP 7.4.33 Development Server (http://127.0.0.1:8000) started
```

When we go on the link given by it we will move to the
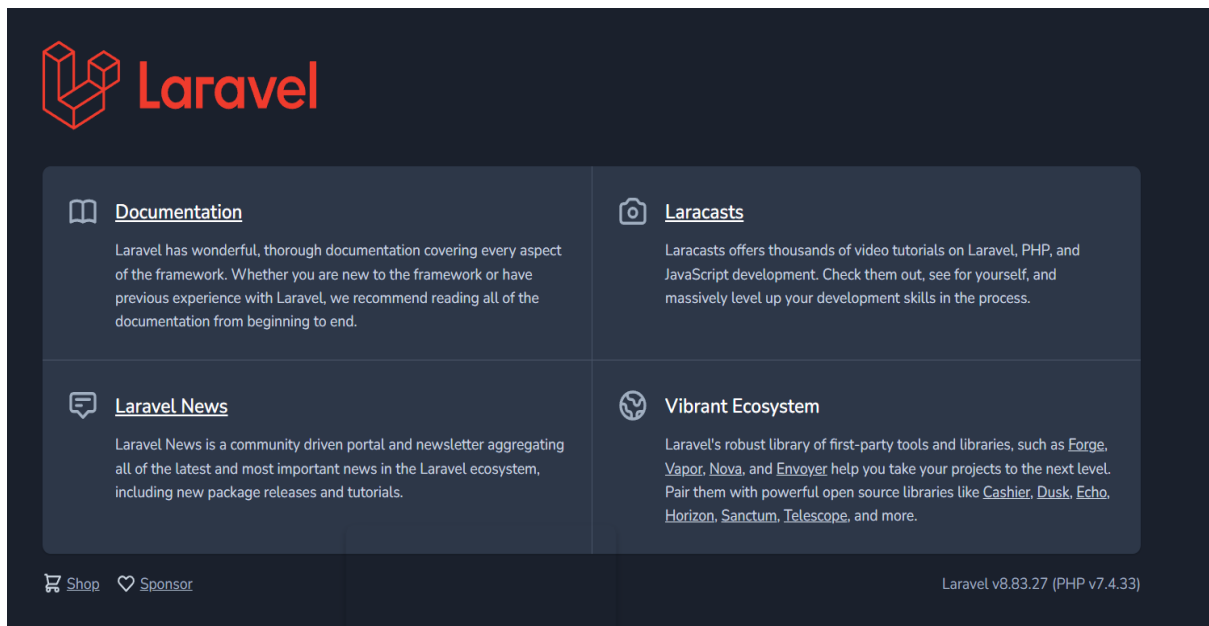
Run Migration

php artisan migrate
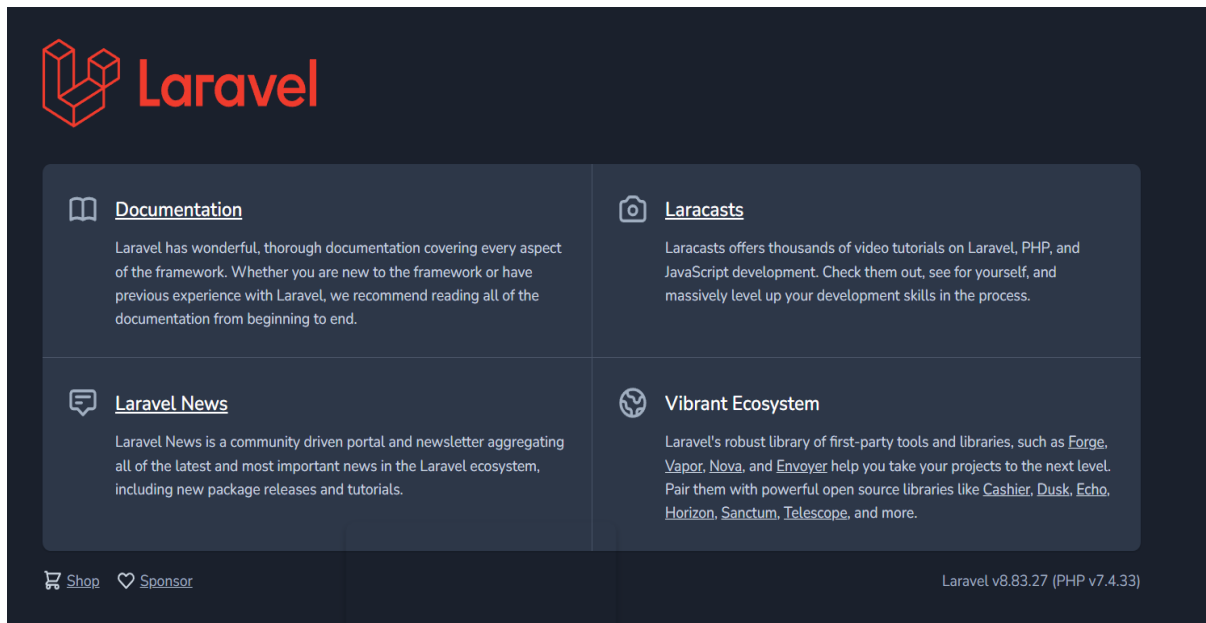
After this we will done with our data base migration task.

Practical 9: To construct the any simple web application using Laravel Framework.

First we are creating our project with these commands

composer create-project laravel/laravel first-project

cd first-project

php artisan serve

First of all we will make a migration in it

php artisan make:migration customers

After that we write code in it in this way in our migration file.

```
public function up()
  {
    Schema::create('customers', function (Blueprint $table) {
      $table->id();
      $table->string('name',60);
      $table->string('email',100);
      $table->enum('gender',["M","F","O"]);
      $table->text('address');
      $table->date('dob');
      $table->timestamps();
    });
```

```
    }
```

After that we will run our server

Then hit the command

php artisan migrate

```
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table (227.68ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated:  2019_08_19_000000_create_failed_jobs_table (156.69ms)
Migrating: 2019_12_14_000001_create_personal_access_tokens_table
Migrated:  2019_12_14_000001_create_personal_access_tokens_table (235.90ms)
Migrating: 2023_11_01_140513_create_customers_table
Migrated:  2023_11_01_140513_create_customers_table (25.10ms)
PS C:\Users\princ\Documents\laravel\first-project> 
```

After that we will write this code in our web.php file

```
Route::get('/', function () {
    return view('welcome');
});
Route::get('/users', [UserController::class, 'show']);
```

After that this will be the code that will be written in our view

```
<!-- resources/views/users.blade.php -->

<style>
    table {
        border-collapse: collapse;
        width: 100%;
    }

    th, td {
        border: 1px solid black;
        padding: 8px;
        text-align: left;
```

```
    }

    th {
      background-color: #f2f2f2;
    }

    tr:nth-child(even) {
      background-color: #f9f9f9;
    }
</style>

<h1>User List</h1>

<table>
  <thead>
    <tr>
      <th>ID</th>
      <th>Name</th>
      <th>Email</th>
      <th>Gender</th>
      <th>Address</th>
      <th>DOB</th>
    </tr>
  </thead>
  <tbody>
    @foreach($data as $user)
      <tr>
        <td>{{ $user->id }}</td>
        <td>{{ $user->name }}</td>
```

```
        <td>{{ $user->email }}</td>

        <td>{{ $user->gender }}</td>

        <td>{{ $user->address }}</td>

        <td>{{ $user->dob }}</td>

      </tr>

    @endforeach

  </tbody>

</table>
```

Then After we have move to the
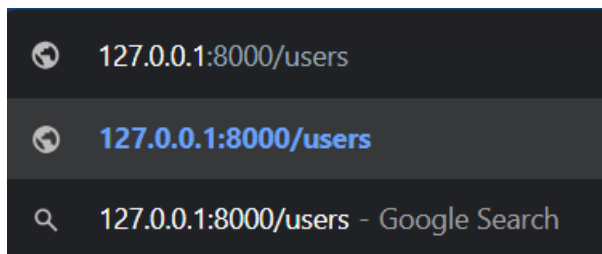
After that we will get this interface

This is our Basic application in Laravel.