

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

Bootstrap is a popular open-source front-end framework developed by Twitter. It simplifies the process of designing and building responsive and mobile-friendly websites. Bootstrap provides a collection of pre-built HTML, CSS, and JavaScript components that you can easily integrate into your web projects.

➤ Versions of Bootstrap

- **Bootstrap 1.0:** The initial release of Bootstrap, known as Twitter Blueprint at the time, was introduced in August 2011. This version laid the foundation for the framework.
- **Bootstrap 2.0:** Released in January 2012, Bootstrap 2.0 brought significant improvements and refined the framework's structure, making it more popular among developers.
- **Bootstrap 3.0:** This version, released in August 2013, introduced a mobile-first design approach, which made it responsive by default. Bootstrap 3 included new components and a flat design style.
- **Bootstrap 4.0:** Bootstrap 4 was a major update released in January 2018. It brought a complete overhaul of the framework, making it more powerful, flexible, and compatible with modern web development practices. Bootstrap 4 introduced a switch to Flexbox, improved grid system, and numerous new features.
- **Bootstrap 5.0:** Bootstrap 5 was released in May 2021. This version continued to build on the improvements made in Bootstrap 4. It emphasized a smaller footprint, dropped the jQuery dependency, and introduced several new utility classes.

➤ Bootstrap in our Website :

There are two ways to use Bootstrap in our website :

1. Go to The Website & Download :

<https://getbootstrap.com/docs/5.0/getting-started/download/>

Compiled CSS and JS

Download ready-to-use compiled code for **Bootstrap v5.0.2** to easily drop into your project, which includes:

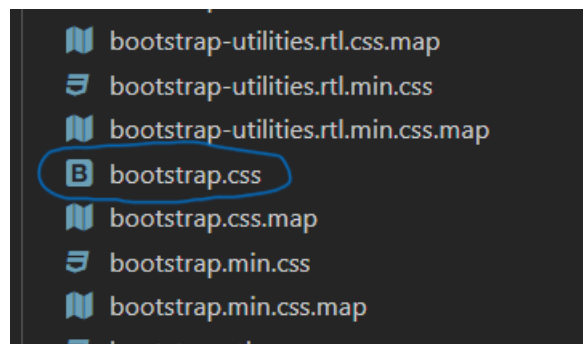
- Compiled and minified CSS bundles (see [CSS files comparison](#))
- Compiled and minified JavaScript plugins (see [JS files comparison](#))

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

[Download](#)

Include it In your workspace & make sure your link is correct while accessing that file.

For this file hierarchy, this must be the link to access 'bootstrap.css' file



```
<link rel="stylesheet" href="bootstrap-5.0.2-dist\css\bootstrap.css">
```

2. Use the Content Delivery Network (CDN) version of Bootstrap in your website

```
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kadq2F9CUG65"
crossorigin="anonymous">
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-
kenU1KFdBIe4zVF0s0G1M5b4hcxpyD9F7jL+jjXkk+Q2h455rYXK/7HAuoJl+0I4"crossor
igin="anonymous"></script>
```

Include this link in head tag of your web page and script into the bottom of the body tag.


➤ Compiled CSS & JS files of Bootstrap

A minified version of Bootstrap files, whether it's CSS or JavaScript, is a compressed and optimized version of the original source code. The primary


purpose of minification is to reduce the file size while preserving the functionality of the code.

These are those minified versions of Bootstrap files

CSS File :

 bootstrap.min.css

Javascript File :

 bootstrap.min.js

```
bootstrap/
├── css/
│   ├── bootstrap.css
│   ├── bootstrap.css.map
│   ├── bootstrap.min.css
│   ├── bootstrap.min.css.map
│   ├── bootstrap-theme.css
│   ├── bootstrap-theme.css.map
│   ├── bootstrap-theme.min.css
│   └── bootstrap-theme.min.css.map
├── js/
│   ├── bootstrap.js
│   └── bootstrap.min.js
└── fonts/
    ├── glyphicons-halflings-regular.eot
    ├── glyphicons-halflings-regular.svg
    ├── glyphicons-halflings-regular.ttf
    ├── glyphicons-halflings-regular.woff
    └── glyphicons-halflings-regular.woff2
```

➤ Basic HTML elements

- Table In Bootstrap

Code

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="bootstrap-5.0.2-dist\css\bootstrap.css">
  <title>Document</title>
</head>
<body>
  <div class="container border border-dark fs-5">
    <div class="row font-weight-bold border border-dark">
      <div class="col border border-dark">No.</div>
      <div class="col border border-dark">Fruit</div>
```

```

        <div class="col border border-dark">Season</div>
    </div>
    <div class="row col border border-dark">
        <div class="col border border-dark">1</div>
        <div class="col border border-dark">Mango 🥭 </div>
        <div class="col border border-dark">Summer</div>
    </div>
    <div class="row col border border-dark">
        <div class="col border border-dark">2</div>
        <div class="col border border-dark">Apple 🍏 </div>
        <div class="col border border-dark">Winter</div>
    </div>
    <div class="row col border border-dark">
        <div class="col border border-dark">3</div>
        <div class="col border border-dark">Banana 🍌 </div>
        <div class="col border border-dark">Any</div>
    </div>
</div>
</body>
</html>

```

Output

No.	Fruit	Season
1	Mango 🥭	Summer
2	Apple 🍏	Winter
3	Banana 🍌	Any

- Form in Bootstrap

```

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

    <link rel="stylesheet" href="bootstrap-5.0.2-
dist\css\bootstrap.css">

    <title>Form</title>

</head>

<body>

    <div class="container">

        <h1>Feedback Form : </h1>

        <div class="form-group row mt-3">

            <label class="col-2 col-form-label">Name : </label>

            <div class="col-10">

```

```

        <input class="form-control" type="text" placeholder="Enter
your name">
    </div>
</div>
<div class="form-group row mt-3">
    <label class="col-2 col-form-label">Class : </label>
    <div class="col-10">
        <input class="form-control" type="text" placeholder="Enter
your class">
    </div>
</div>
<div class="form-group row mt-3">
    <label class="col-2 col-form-label">Message : </label>
    <div class="col-10">
        <textarea class="form-control" rows="4" placeholder="Enter
your Message"></textarea>
    </div>
    <div class="form-group row mt-3">
        <div class="offset-2">
            <button class="btn bg-primary">Submit</button>
        </div>
    </div>
</div>
</div>
</body>
</html>

```

Output :

Feedback Form :

Name :	<input type="text" value="Enter your name"/>
Class :	<input type="text" value="Enter your class"/>
Message :	<div><input type="text" value="Enter your Message"/></div>
	<input type="button" value="Submit"/>

Experiment 2 : To understand and deploy the multicolumn grid layout of Bootstrap.

➤ Breakpoints

Breakpoints are customizable widths that determine how your responsive layout behaves across device or viewport sizes in Bootstrap.

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

Code

```
<!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="bootstrap-5.0.2-dist\css\bootstrap.css">
</head>
<body>
  <div class="container">
    <h1>Multi-Column Grid Layout</h1>
    <div class="row">
      <div class="col-lg-3 col-md-4 col-6">
        <div class="card">
          <div class="card-body">
            <h3>Column 1</h3>
```

<p>Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ab vitae at in suscipit quod maxime dignissimos error architecto dolore voluptatum?</p>

</div>

</div>

</div>

<div class="col-lg-3 col-md-4 col-6">

<div class="card">

<div class="card-body">

<h3>Column 2</h3>

<p>Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ab vitae at in suscipit quod maxime dignissimos error architecto dolore voluptatum?</p>

</div>

</div>

</div>

<div class="col-lg-3 col-md-4 col-6">

<div class="card">

<div class="card-body">

<h3>Column 3</h3>

<p>Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ab vitae at in suscipit quod maxime dignissimos error architecto dolore voluptatum?</p>

</div>

</div>

</div>

<div class="col-lg-3 col-md-4 col-6">

<div class="card">

<div class="card-body">

<h3>Column 4</h3>

<p>Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ab vitae at in suscipit quod maxime dignissimos error architecto dolore voluptatum?</p>

</div>

</div>

</div>

</div>

</div>

</body>

</html>

Output

- For large Screen size there are 4 columns

Multi-Column Grid Layout

Column 1 Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ab vitae at in suscipit quod maxime dignissimos error architecto dolore voluptatum?	Column 2 Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ab vitae at in suscipit quod maxime dignissimos error architecto dolore voluptatum?	Column 3 Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ab vitae at in suscipit quod maxime dignissimos error architecto dolore voluptatum?	Column 4 Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ab vitae at in suscipit quod maxime dignissimos error architecto dolore voluptatum?
---	---	---	---

- For medium Screen there are 3 columns

Multi-Column Grid Layout

Column 1 Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ab vitae at in suscipit quod maxime dignissimos error architecto dolore voluptatum?	Column 2 Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ab vitae at in suscipit quod maxime dignissimos error architecto dolore voluptatum?	Column 3 Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ab vitae at in suscipit quod maxime dignissimos error architecto dolore voluptatum?
Column 4 Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ab vitae at in suscipit quod maxime dignissimos error architecto dolore voluptatum?		

- For other Screen size there are 2 columns

Multi-Column Grid Layout

Column 1 Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ab vitae at in suscipit quod maxime dignissimos error architecto dolore voluptatum?	Column 2 Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ab vitae at in suscipit quod maxime dignissimos error architecto dolore voluptatum?
Column 3 Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ab vitae at in suscipit quod maxime dignissimos error architecto dolore voluptatum?	Column 4 Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ab vitae at in suscipit quod maxime dignissimos error architecto dolore voluptatum?

Experiment: 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

- A computer with a text editor and web browser
- 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>
```

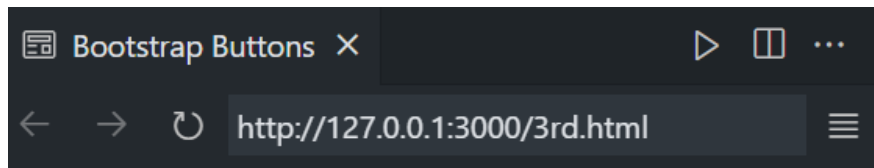
```
<script  
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
```

Step 2: Deploying Buttons

1. Create buttons using Bootstrap classes. For example:

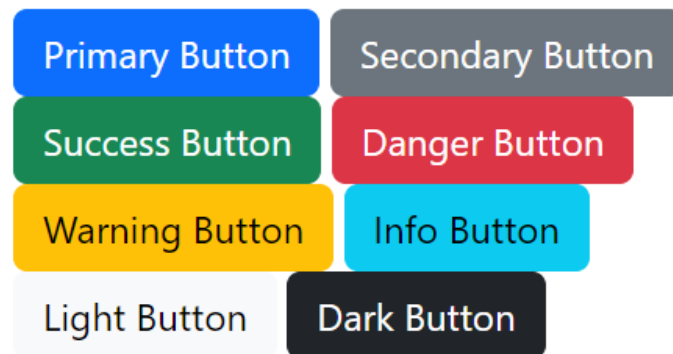
```
<button class="btn btn-primary">Primary Button</button>  
<button class="btn btn-secondary">Secondary Button</button>  
<button class="btn btn-success">Success Button</button>
```

2. Customize the buttons by experimenting with different classes and styles provided by Bootstrap.



Bootstrap Buttons

Click on the buttons to see different styles.



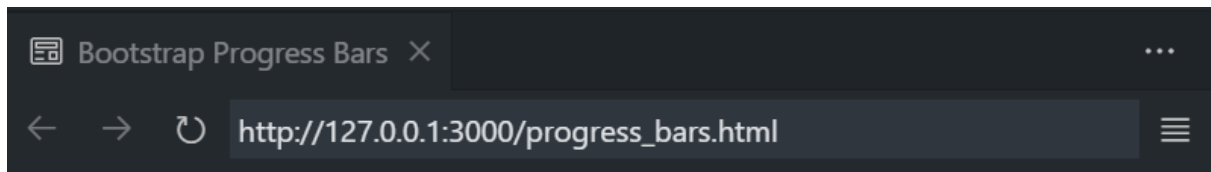
Step 3: Creating Progress Bars

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.



Bootstrap Progress Bars

Progress bars with different contextual types:



Step 4: Designing Modals

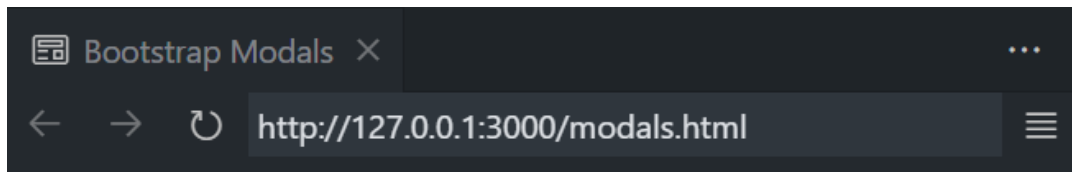
1. Create a modal dialog using Bootstrap. This modal can display additional information or perform specific actions when triggered. Example modal code:

```
<!-- 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>
```

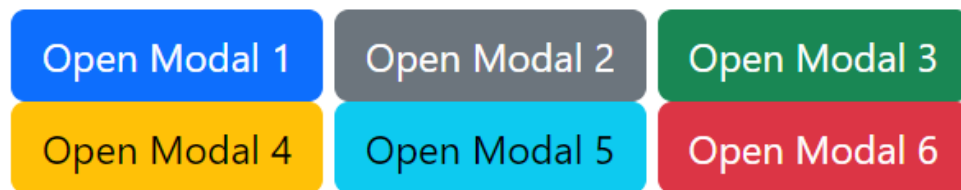
...

2. Customize the modal's content and appearance based on your project's requirements.



Bootstrap Modals

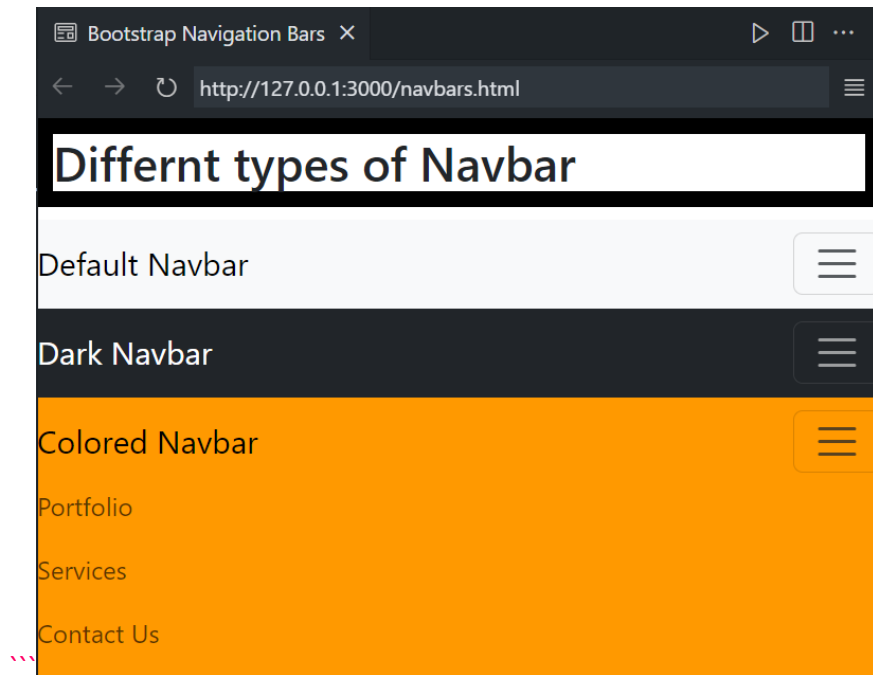
Click on the buttons to open different modals.



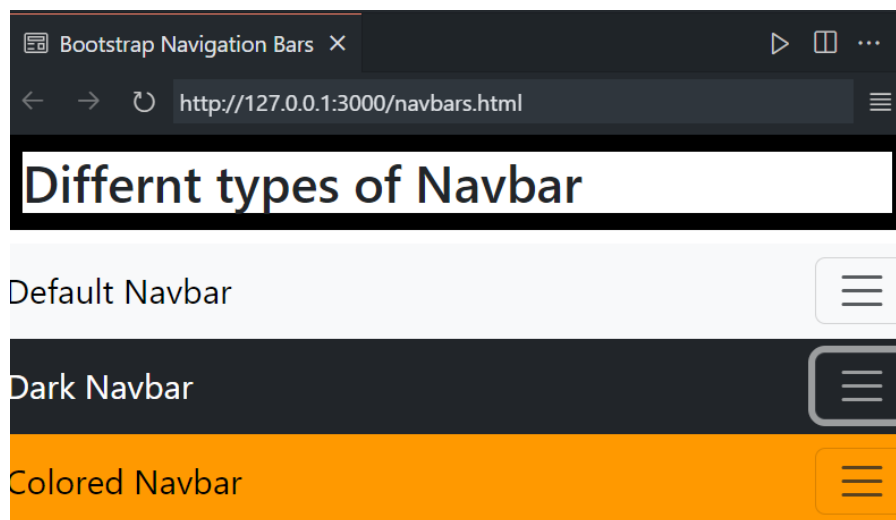
Step 5: Implementing Navigation Bars

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>
```



2. Customize the navigation bar by adding menu items and adjusting the appearance to match your application's branding.



Conclusion

In this practical exercise, we have successfully deployed different Bootstrap components, including buttons, progress bars, modals, and navigation bars. Bootstrap simplifies the process of creating visually appealing and responsive web applications, enhancing the overall user experience. These components are highly customizable, allowing developers to tailor them to

specific project requirements. By mastering these Bootstrap elements, you can create interactive and engaging web applications.

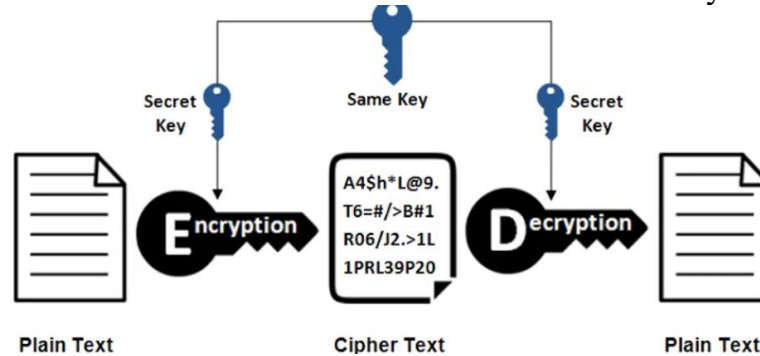
References

- Bootstrap Documentation: <https://getbootstrap.com/docs/5.3/getting-started/introduction/>

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"
```

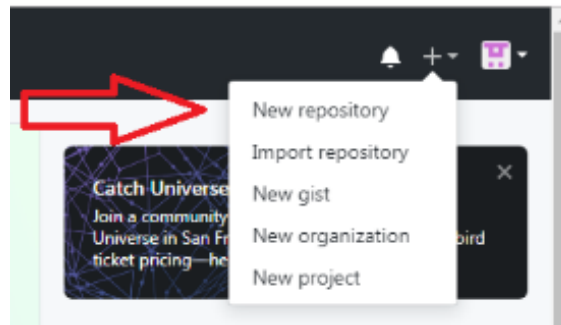
2. **Add your SSH Key to the SSH Agent:** You'll want to add your private key to the SSH agent to manage your keys. You can do this with the following command

```
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
```

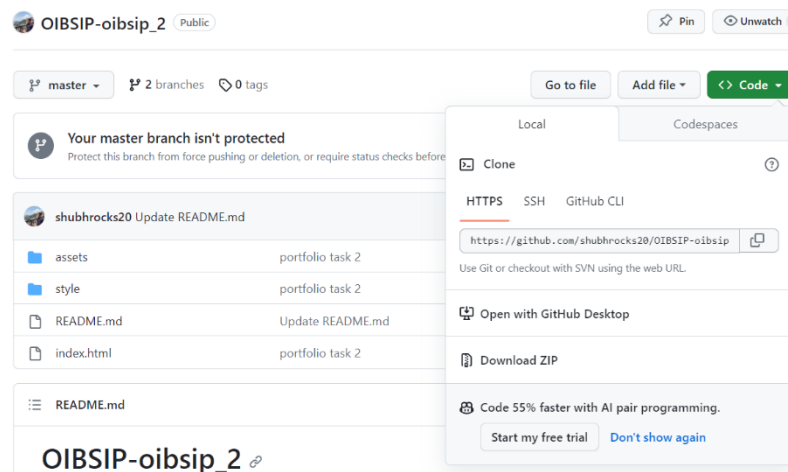
When you push, Git uploads your commits to the remote repository, making your changes available to others who can then pull those changes into their own local repositories. This process is a central part of collaboration and keeping all contributors in sync.

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](https://codeigniter.com/) 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 :)

[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.

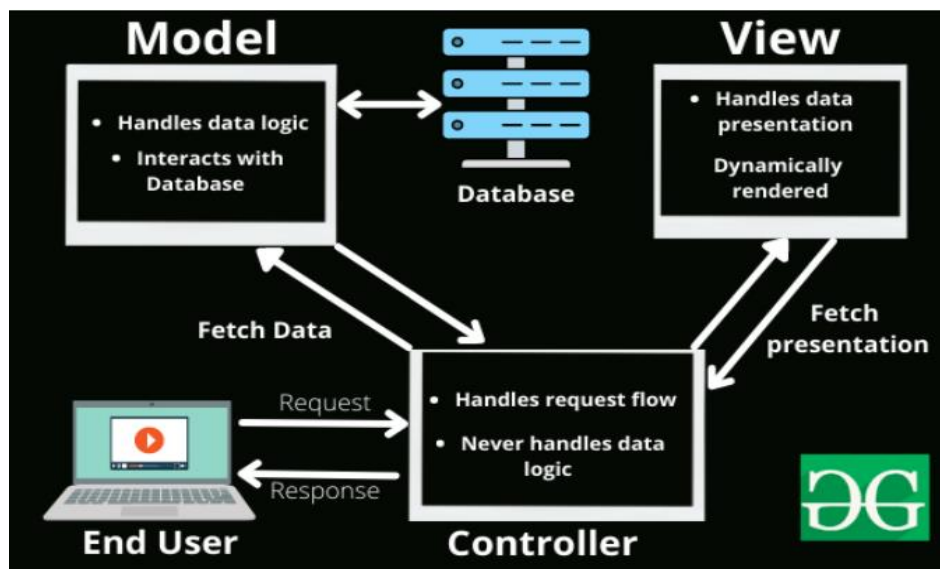
2. Characteristics:

- **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.

EXPERIMENT 7: To construct a simple login page web application to authenticate users using CodeIgniter Framework and also perform CRUD 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. create.php :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CRUD CREATE IN CI</title>
  <link rel="stylesheet"
href="http://localhost/crud_in_ci/public/css/bootstrap.min.css">
</head>
<body>
  <div class="container bg-warning mt-5 pt-3 pb-3" >
    <h1 style="text-align:center">CREATE OPERATION</h1>
  </div>

  <div class="container mt-5">
    <form method="POST" action="<?php echo site_url('Crud/add'); ?>">
    <div class="form-group">
      <label for="urn">URN</label>
      <input type="text" name="urn" id="urn" class="form-control"
placeholder="eg. 212134" required>
    </div>

    <div class="form-group mt-3">
      <label for="name">Name</label>
```



```

        <input type="text" name="name" id="name" class="form-control"
placeholder="eg. xyz" required>
    </div>

    <div class="form-group mt-3">
        <label for="branch">Branch</label>
        <input type="text" name="branch" id="branch" class="form-control"
placeholder="eg. cse" required>
    </div>

    <button type="submit" class="btn btn-primary mt-3">Submit</button>
</form>
</div>

</body>
</html>

```

2. edit.php:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CRUD UPDATE IN CI</title>
    <link rel="stylesheet"
href="http://localhost/crud_in_ci/public/css/bootstrap.min.css">
</head>
<body>
    <div class="container bg-warning mt-5 pt-3 pb-3" >
        <h1 style="text-align:center">UPDATE OPERATION</h1>
    </div>

    <div class="container mt-5">
        <form method="POST" action="<?= site_url('Crud/update'); ?>">
        <div class="form-group">
            <label for="urn">URN</label>
            <input type="text" name="urn" id="urn" class="form-control"
placeholder="eg. 212134" value="<?php echo $student['urn']; ?>">
        </div>

        <div class="form-group mt-3">
            <label for="name">Name</label>
            <input type="text" name="name" id="name" class="form-control"
placeholder="eg. xyz" value="<?php echo $student['name']; ?>">
        </div>

```

```

        <div class="form-group mt-3">
            <label for="branch">Branch</label>
            <input type="text" name="branch" id="branch" class="form-control"
placeholder="eg. cse" value="<?php echo $student['branch']; ?>">
        </div>

        <button type="submit" class="btn btn-primary mt-3">Submit</button>
    </form>
</div>

</body>
</html>

```

3.crud.php:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CRUD IN CI</title>
    <link rel="stylesheet"
href="http://localhost/crud_in_ci/public/css/bootstrap.min.css">
</head>
<body>
    <div class="container bg-warning mt-5 pt-3 pb-3" >
        <h1 style="text-align:center">CRUD OPERATIONS IN CODEIGNITER</h1>
    </div>

    <div class="container mt-4">
        <div class="clear-fix">
            <h3 style="float:left">All Students</h3>
            <a class="btn btn-primary" style="float:right"
href="http://localhost/crud_in_ci/public/create">Add New Student</a>
        </div>

        <table class="table table-striped table-hover">
            <thead>
                <tr>
                    <th>Student URN</th>
                    <th>Student Name</th>
                    <th>Student Branch</th>
                    <th>Operations</th>
                </tr>
            </thead>

```

```

        <tbody>
            <?php if($students) : ?>
            <?php foreach($students as $row) : ?>
            <tr>
                <td><?php echo $row['urn']; ?></td>
                <td><?php echo $row['name']; ?></td>
                <td><?php echo $row['branch']; ?></td>
                <td>
                    <a class="btn btn-success" href="<?=
site_url('Crud/edit/' . $row['urn']) ?>">Edit</a>
                    <a class="btn btn-danger" href="<?=
site_url('Crud/delete/' . $row['urn']) ?>">Delete</a>
                </td>
            </tr>
            <?php endforeach; ?>
            <?php endif; ?>
        </tbody>
    </table>

</div>

</body>
</html>

```

Models:

1.StudentModel.php:

```

<?php

namespace App\Models;

use CodeIgniter\Model;

class StudentModel extends Model{
    protected $table = 'students';
    protected $primaryKey = 'urn';
    protected $allowedFields = ['urn', 'name', 'branch'];
}

?>

```

Controller:

1.Crud.php

```
<?php

namespace App\Controllers;
use App\Models\StudentModel;

class Crud extends BaseController
{
    public function index(){
        $model = new StudentModel();
        $data['students'] = $model->findAll();
        return view('crud', $data);
    }

    public function create(){
        return view('create');
    }

    public function add(){
        $model = new StudentModel();
        $data = [
            'urn' => $this->request->getPost('urn'),
            'name' => $this->request->getPost('name'),
            'branch' => $this->request->getPost('branch')
        ];

        $model->insert($data);
        return redirect('Crud');
    }

    public function edit($id = null){
        $model = new StudentModel();
        $data['student'] = $model->where('urn',$id)->first();
        return view('edit',$data);
    }

    public function update(){
        $model = new StudentModel();
        $data = [
            'urn' => $this->request->getPost('urn'),
            'name' => $this->request->getPost('name'),
            'branch' => $this->request->getPost('branch')
        ];
        $id = $this->request->getPost('urn');
        $model->update($id,$data);
    }
}
```

```

        return redirect('Crud');
    }

    public function delete($id = null){
        $model = new StudentModel();
        $data['user'] = $model->where('urn', $id)->delete();
        return redirect('Crud');
    }
}

```

Create the routes:

```

<?php

use CodeIgniter\Router\RouteCollection;

$routees->get('/', 'Crud::index');
$routees->get('/create', 'Crud::create');
$routees->post('/Crud/add', 'Crud::add');
$routees->get('/Crud/edit/(:num)', 'Crud::edit/$1');
$routees->post('/Crud/update/', 'Crud::update');
$routees->get('/Crud/delete/(:num)', 'Crud::delete/$1');

```

In Database.php add the database:

```

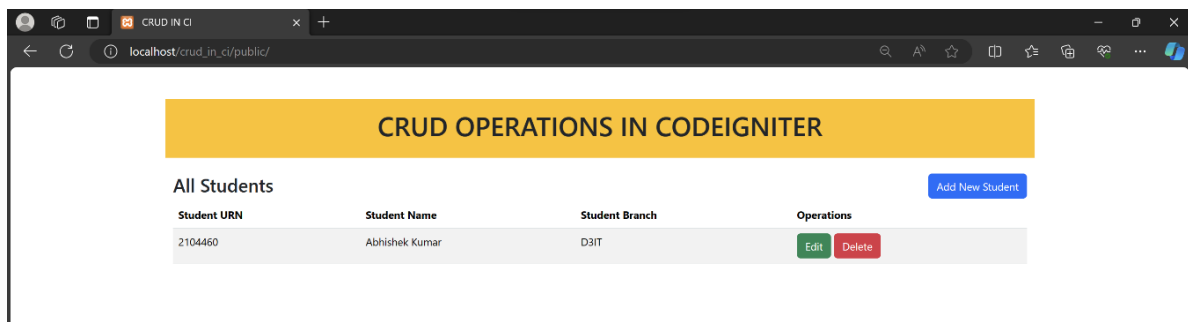
'hostname'      => 'localhost',
'username'      => 'root',
'password'      => '',
'database'      => 'crud in ci',

```

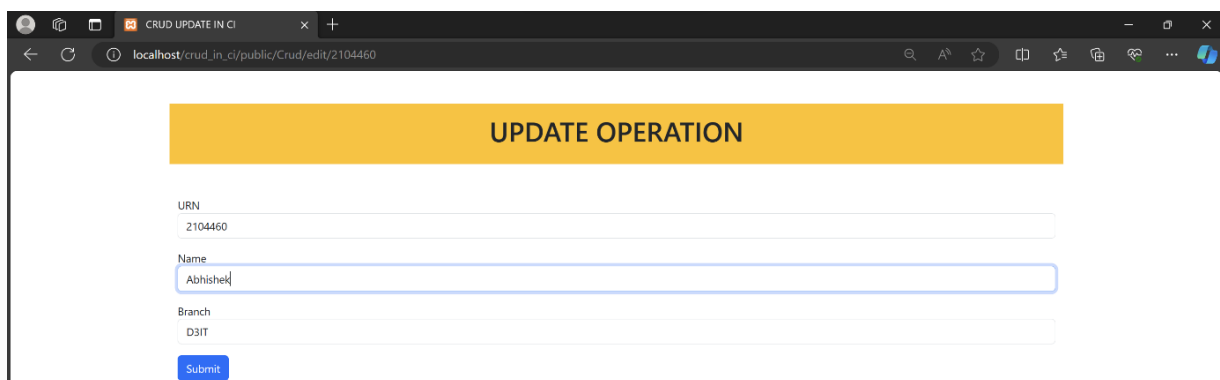
OUTPUT : Add Users :

The screenshot shows a web browser window with the address bar displaying 'localhost/crud_in_ci/public/create'. The page has a yellow header bar with the text 'CREATE OPERATION'. Below the header, there are three input fields: 'URN' with a placeholder 'eg. 212134', 'Name' with a placeholder 'eg. xyz', and 'Branch' with a placeholder 'eg. cse'. At the bottom of the form is a blue button labeled 'Submit'.

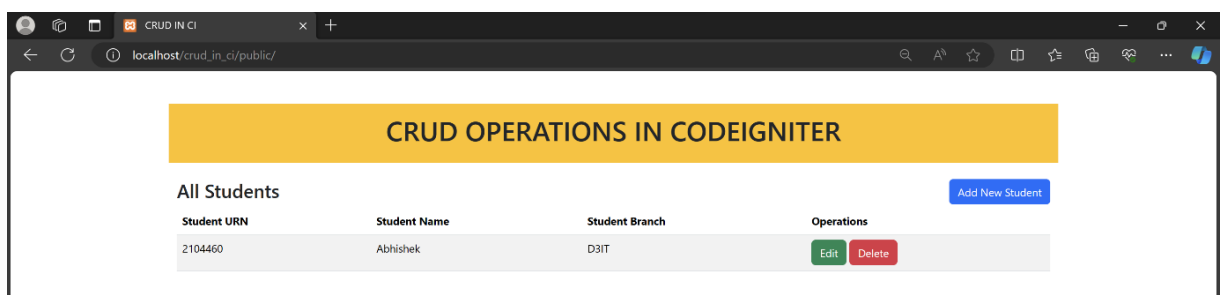
User View:



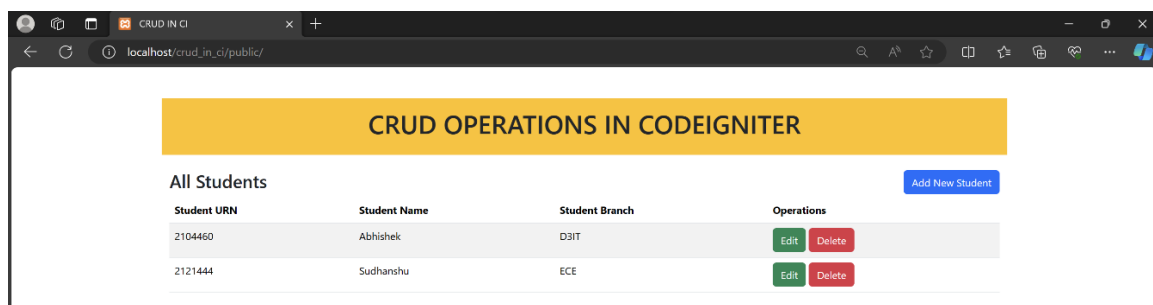
Update data:



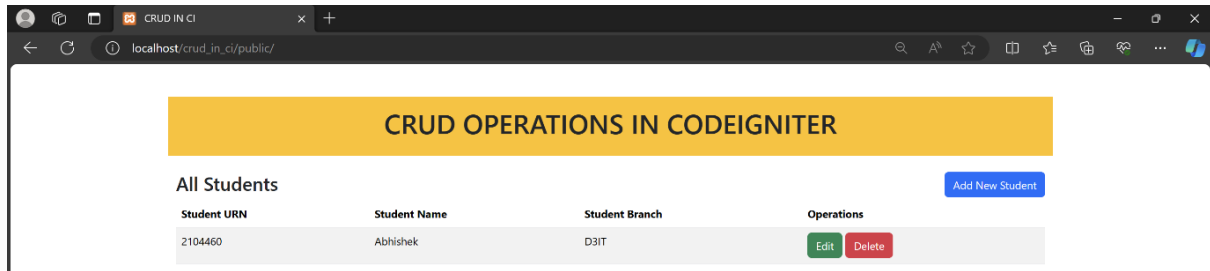
After Updation :



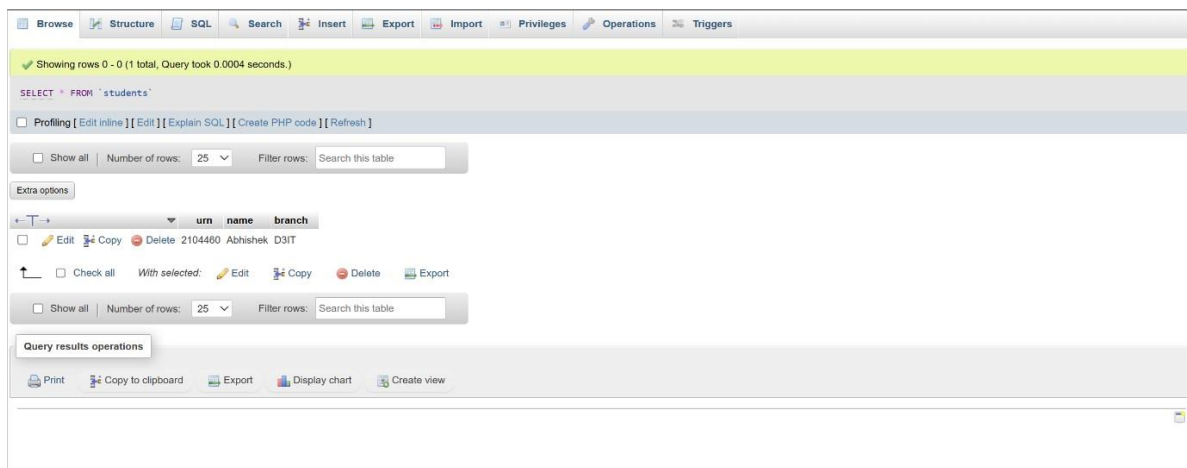
After Adding New User :



Deleted the users:



Database :



Experiment 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.

[illegible]

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
```

```
Package manifest generated successfully.
78 packages you are using are looking for funding.
Use the 'composer fund' command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force
No publishable resources for tag [laravel-assets].
Publishing complete.
No security vulnerability advisories found.
> @php artisan key:generate --ansi
Application key set successfully.
PS C:\Users\princ\documents\laravel>
```

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

```
cd myProject
```

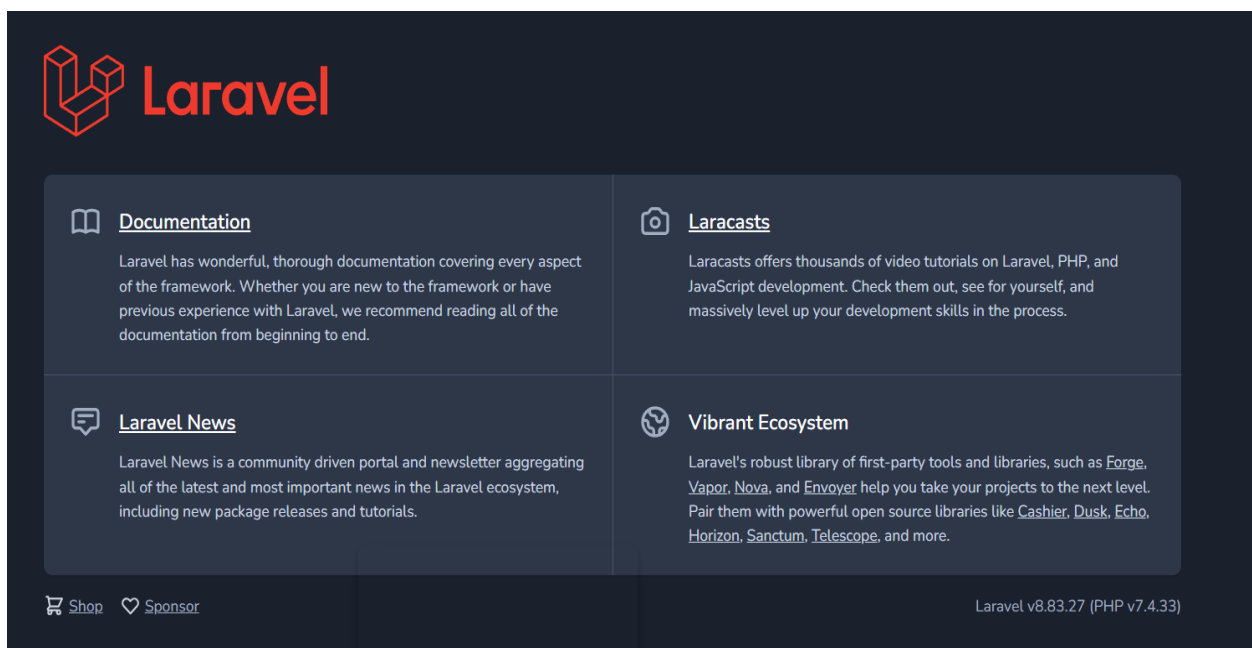
After this we will move in our Project directory.

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

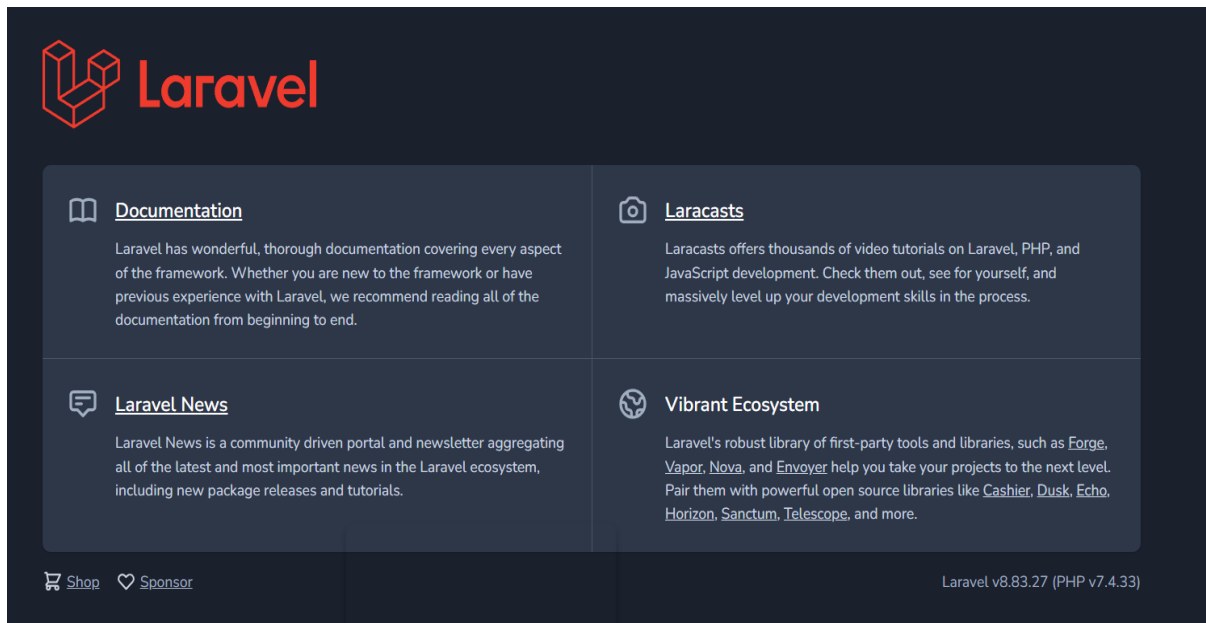
php artisan migrate

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

Experiment 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>

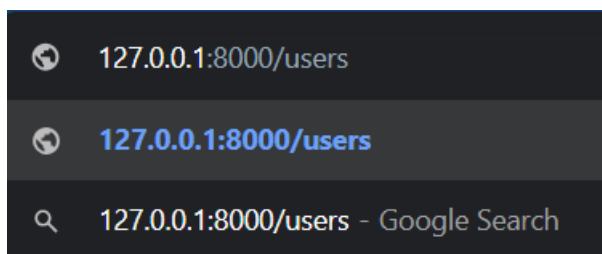
    @endforeach

</tbody>

</table>

```

Then After we have move to the



After that we will get this interface



User List

ID	Name	Email	Gender	Address	DOB
1	John Doe	john.doe@example.com	M	123 Main St, Cityville	1990-05-15
2	Jane Smith	jane.smith@example.com	F	456 Oak St, Townsville	1985-08-22
3	Bob Johnson	bob.johnson@example.com	M	789 Pine St, Villagetown	1982-11-10
4	Alice Williams	alice.williams@example.com	F	101 Cedar St, Hamletville	1993-02-18
5	Charlie Brown	charlie.brown@example.com	O	202 Birch St, Countryside	1987-07-03
6	Eva Davis	eva.davis@example.com	F	303 Elm St, Riverside	1995-04-29
7	David Miller	david.miller@example.com	M	404 Maple St, Suburbia	1980-09-12
8	Sophie Wilson	sophie.wilson@example.com	F	505 Walnut St, Uptown	1998-12-08
9	Ryan Thomas	ryan.thomas@example.com	M	606 Pineapple St, Tropicville	1989-06-25
10	Olivia Lee	olivia.lee@example.com	F	707 Peach St, Orchardtown	1997-03-14

This is our Basic application in Laravel.