GURU NANAK DEV ENGINEERING COLLEGE

DEPARTMENT OF
INFORMATION TECHNOLOGY

PRACTICAL FILE: Advanced Web Technologies
(5th Semester 2021-2025 Batch)
JULY-DECEMBER,2023

Submitted to:
(Prof).Akshay Girdhar

Submitted by:
Harjot Singh
URN:2104507
CRN:2121130
D3IT(B2)

INDEX

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

Introduction to Bootstrap:

Bootstrap is a free front-end framework for faster and easier web development.Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins.Bootstrap also gives you the ability to easily create responsive designs.

Bootstrap History:

Bootstrap was developed by Mark Otto and Jacob Thornton at Twitter, and released as an open source product in August 2011 on GitHub.

Why Use Bootstrap?

Advantages of Bootstrap:

- **Easy to use:** Anybody with just basic knowledge of HTML and CSS can start using Bootstrap
- **Responsive features:** Bootstrap's responsive CSS adjusts to phones, tablets, and desktops
- **Mobile-first approach:** In Bootstrap 3, mobile-first styles are part of the core framework
- **Browser compatibility:** Bootstrap is compatible with all modern browsers (Chrome, Firefox, Internet Explorer, Edge, Safari, and Opera)

Bootstrap Versions

The Bootstrap 3, which was released in 2013. However, the newer versions; Bootstrap 4 (released 2018) and Bootstrap 5 (released 2021).Bootstrap 5 is the newest version of Bootstrap; with new components, faster stylesheets, more responsiveness etc. It supports the latest, stable releases of all major browsers and platforms. However, Internet Explorer 11 and down is not supported.The main differences between Bootstrap 5 and Bootstrap 3 & 4, is that Bootstrap 5 has switched to JavaScript instead of jQuery.

Where to Get Bootstrap?

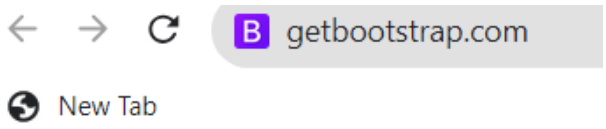
There are two ways to start using Bootstrap on your own web site.We can:

- Download Bootstrap from getbootstrap.com
- Include Bootstrap from a CDN

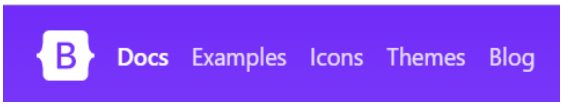
Downloading Bootstrap:

If we want to download and host Bootstrap yourself, go to getbootstrap.com, and follow the instructions there.

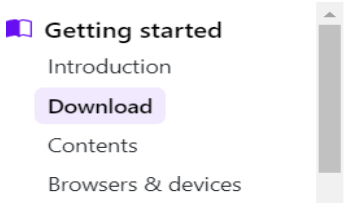
1. **Open the Bootstrap website in your internet browser.** Type https://getbootstrap.com into the address bar, and press Enter on your keyboard.



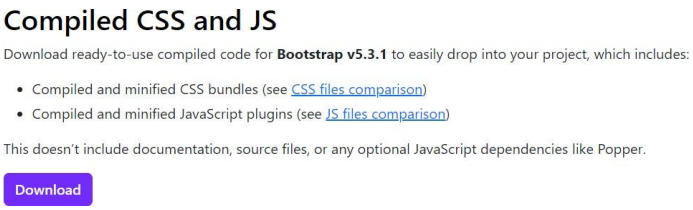
2. **Click the Docs button.** This will open the "Docs" page.



3. **Click the Download button.** This will open the "Download" page.



4. **Click the Download button below "Compiled CSS and JS."** This will download the complete Bootstrap files to your computer as a ZIP archive.



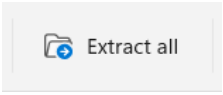
5. **Extract the files from the ZIP archive.** Find the ZIP file you just downloaded, and double-click on it to extract all the files in it to the same folder.

This will extract two folders named "**css**" and "**js**."

If your unzipper app does not automatically extract the files, check out this article to see all the ways you can export a ZIP archive.

6. **Move the extracted folders to the same folder as your website HTML files.** Open the folder that contains all your website's HTML files, and drag the "**css**" and "**js**" folders here to move them to the same folder as your website documents.

You can now link the files to your HTML files, and start using Bootstrap in your code.



Bootstrap CDN: If we don't want to download and host Bootstrap yourself, we can include it from a CDN (Content Delivery Network).

MaxCDN provides CDN support for Bootstrap's CSS and JavaScript. You must also include jQuery:

MaxCDN:

<!-- Latest compiled and minified CSS --> <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
<!-- jQuery library --> <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.4/jquery.min.js"></script>
<!-- Latest compiled JavaScript --> <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

CODE:

1. html File Code:

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

<head>
  <title>Bootstrap - Grid</title>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js"></script>
  <link rel="stylesheet" href="style6.css">
</head>

<body>
  <div id="carouselExample" class="carousel slide">
    <div class="carousel-inner">
      <div class="carousel-item active">
        <img src="https://wallpapercave.com/wp/wp6350578.jpg" height="500" width="1250" alt="...">
      </div>
    </div>
```

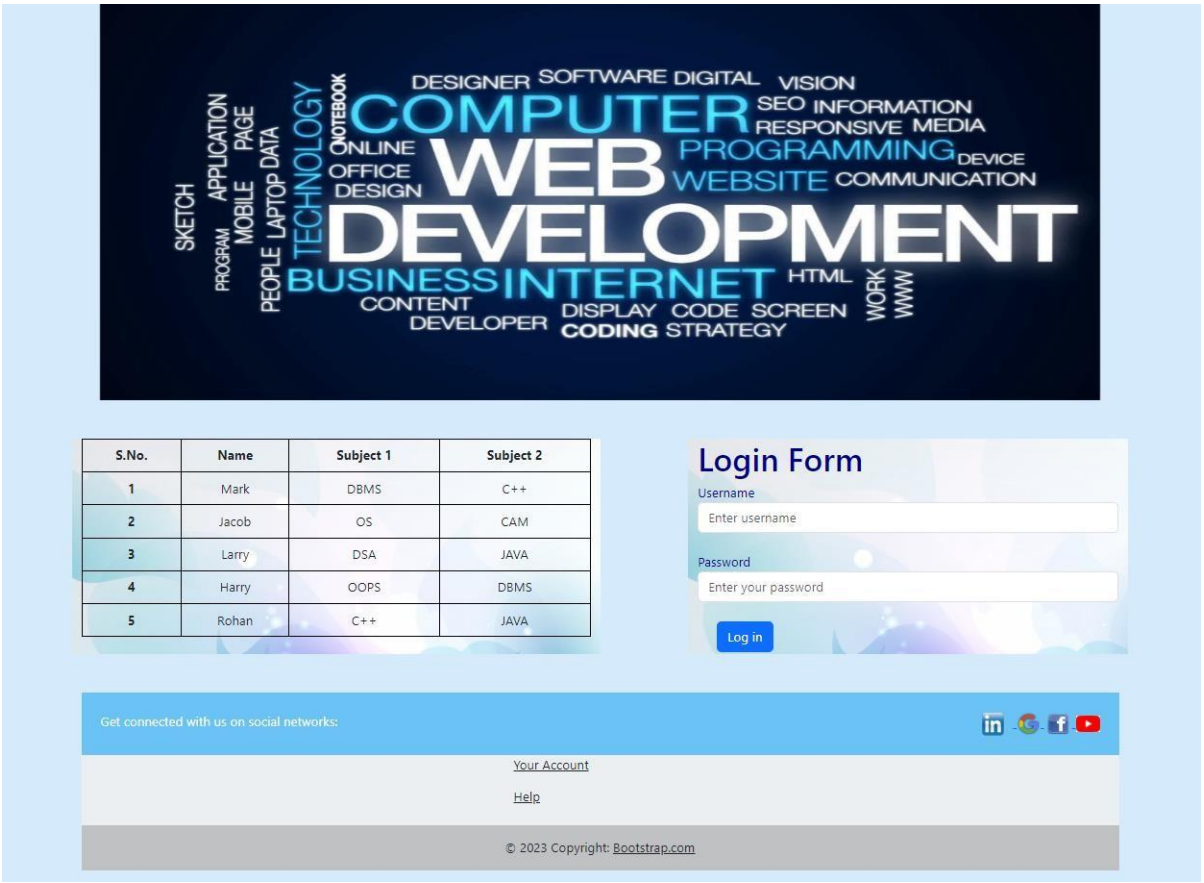
```
</div><br><br>
<div class="container">
   <div class="row">
      <div class="col-md-6">
      <table class="table">
         <thead>
          <tr>
            <th scope="col">S.No.</th>
            <th scope="col">Name</th>
            <th scope="col">Subject 1</th>
            <th scope="col">Subject 2</th>
          </tr>
         </thead>
         <tbody>
          <tr>
            <th scope="row">1</th>
            <td>Mark</td>
            <td>DBMS</td>
            <td>C++</td>
          </tr>
          <tr>
            <th scope="row">2</th>
            <td>Jacob</td>
            <td>OS</td>
            <td>CAM</td>
          </tr>
          <tr>
            <th scope="row">3</th>
            <td>Larry</td>
            <td>DSA</td>
            <td>JAVA</td>
          </tr>
          <tr>
            <th scope="row">4</th>
            <td>Harry</td>
            <td>OOPS</td>
            <td>DBMS</td>
          </tr>
          <tr>
            <th scope="row">5</th>
```

```
            <td>Rohan</td>
            <td>C++</td>
            <td>JAVA</td>
           </tr>
          </tbody>
         </table>
        </div>
        <div class="col-md-1"></div>
      <div class="col-md-5">
        <form>
          <div class="form-group">
            <h1> Login Form</h1>
            <label for="exampleInputUsername1">Username</label>
            <input type="Username" class="form-control" id="exampleInputUsername1" aria-
describedby="UsernameHelp"
              placeholder="Enter username"><br>
            <small id="UsernameHelp" class="form-text text-muted"></small>
          </div>
          <div class="form-group">
            <label for="exampleInputPassword1">Password</label>
            <input type="password" class="form-control" id="exampleInputPassword1"
              placeholder="Enter your password">
          </div><br>
          <div class="form-check">
            <button type="submit" class="btn btn-primary">Log in</button><br>
          </div>
        </div>
      </div>
      </div>
      </div>
      <!-- Remove the container if you want to extend the Footer to full width. -->
<div class="container my-5">
  <!-- Footer -->
  <footer
      class="text-center text-lg-start text-dark"
      style="background-color: #ECEFF1"
      >
    <!-- Section: Social media -->
    <section
        class="d-flex justify-content-between p-4 text-white"
```

```
          style="background-color: #6bc2f5"
          >
<div class="me-5">
        <span>Get connected with us on social networks:</span>
      </div>
      <div id="socialIcons">
        <a href="https://www.linkedin.com/" title="Share on LinkedIn">
          <img src="https://logosmarcas.net/wp-content/uploads/2020/03/LinkedIn-Emblema.png" height="30" width="50"alt="LinkedIn Logo">
        </a>
        <a href="https://www.google.com" title="Visit Google">
          <img src="https://www.pngmart.com/files/16/Google-Logo-PNG-Image.png" height="30" width="30" alt="Google Logo">
        </a>
        <a href="https://www.facebook.com" title="Visit Facebook">
          <img src="https://i0.wp.com/www.bambucoworking.com/wp-content/uploads/2017/04/facebook-logo-png-2335.png?ssl=1" height="30" width="35" alt="Facebook Logo">
        </a>
        <a href="https://www.youtube.com" title="Visit Youtube">
          <img src="https://tse3.mm.bing.net/th?id=OIP.RsawtlednPmK5gnIy6XqIAHaFL&pid=Api&P=0&h=180" height="20" width="30" alt="Youtube Logo">
        </a>

      </div>
    </section>

        <div class="col-md-3 col-lg-2 col-xl-2 mx-auto mb-4">
            <p>
              <a href="#!" class="text-dark">Your Account</a>
            </p>
            <p>
              <a href="#!" class="text-dark">Help</a>
            </p>
        </div>


    <!-- Copyright -->
    <div
      class="text-center p-3"
```

```
        style="background-color: rgba(12, 12, 12, 0.2)"
        >
      © 2023 Copyright:
      <a class="text-dark" >Bootstrap.com</a
        >
    </div>
    <!-- Copyright -->
  </footer>
  <!-- Footer -->
 </div>
 <!-- End of .container -->
</body>

</html>
```

OUTPUT:



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.

Breakpoints are the building blocks of responsive design: Use them to control when your layout can be adapted at a particular viewport or device size.

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
X-Small	*None*	<576px
Small	sm	≥576px
Medium	md	≥768px
Large	lg	≥992px
Extra large	xl	≥1200px
Extra extra large	xxl	≥1400px

CODE:

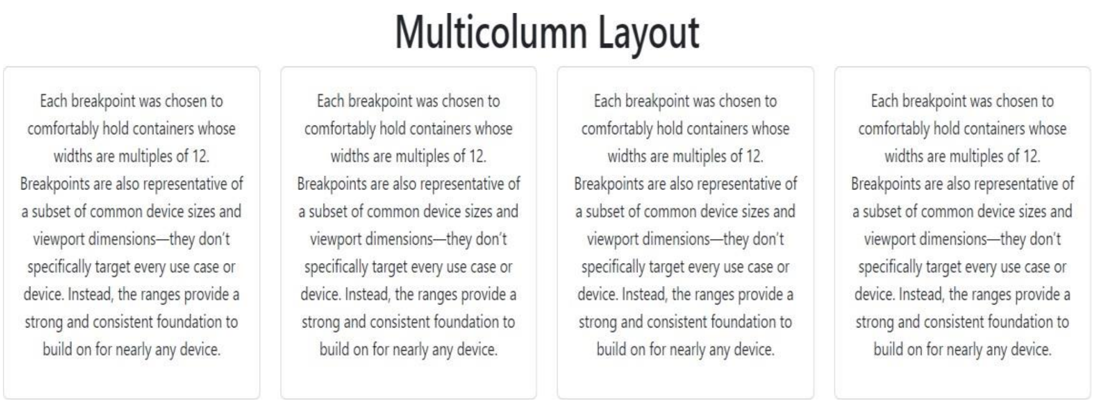
multi.html File Code:

```
<!DOCTYPE html>
  <html lang="en">
  <head>
    <title>Bootstrap - Grid</title>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js"></script>
    <link rel="stylesheet" href="style4.css">
```

```
  </head>
<h1>
  <h1>Multicolumn Layout</h1>
<div class="container ">
  <div class="row ">
    <div class="col-lg-3 md-4 col-6 ">
      <div class="card">
        <div class="card-body">
          <p>Each breakpoint was chosen to comfortably hold containers whose widths are multiples of 12. Breakpoints are also representative of a subset of common device sizes and viewport dimensions—they don’t specifically target every use case or device. Instead, the ranges provide a strong and consistent foundation to build on for nearly any device.</p>
        </div>
      </div>
      </div>
      <div class="col-lg-3 md-4 col-6 ">
        <div class="card">
          <div class="card-body">
            <p>Each breakpoint was chosen to comfortably hold containers whose widths are multiples of 12. Breakpoints are also representative of a subset of common device sizes and viewport dimensions—they don’t specifically target every use case or device. Instead, the ranges provide a strong and consistent foundation to build on for nearly any device.</p>
          </div>
        </div>
        </div>
      <div class="col-lg-3 md-4 col-6 ">
        <div class="card">
          <div class="card-body">
            <p>Each breakpoint was chosen to comfortably hold containers whose widths are multiples of 12. Breakpoints are also representative of a subset of common device sizes and viewport dimensions—they don’t specifically target every use case or device. Instead, the ranges provide a strong and consistent foundation to build on for nearly any device.</p>
          </div>
        </div>
      </div>
      <div class="col-lg-3 md-4 col-6 ">
        <div class="card">
          <div class="card-body">
            <p>Each breakpoint was chosen to comfortably hold containers whose widths are multiples of 12. Breakpoints are also representative of a subset of common device sizes and viewport dimensions—they don’t specifically target every use case or device. Instead, the ranges provide a strong and consistent foundation to build on for nearly any device.</p>
          </div>
```

```
            </div>
          </div>
        </div>
    </div>
</body>
</html>
```

OUTPUT:



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

Bootstrap provides a set of classes and styles for creating buttons in your web applications. These buttons can be used for various purposes, such as links, actions, and form submissions. Here's how to create Bootstrap buttons:

Basic Button Styles:

Bootstrap offers different button styles, including primary, secondary, success, danger, warning, info, and light, which you can apply to your buttons using appropriate classes. Here's an example of how to create basic Bootstrap buttons:

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

Bootstrap provides a simple and customizable way to create progress bars for your web applications. You can use progress bars to show the completion status of a task or to indicate the progress of a file upload, download, or any other process. Here's how to create Bootstrap progress bars:

Basic Progress Bar:

To create a basic Bootstrap progress bar, you can use the .progress and .progress-bar classes. Here's a simple example:

```
<div class="progress">
  <div class="progress-bar" role="progressbar" style="width: 50%;" aria-valuenow="50" aria-valuemin="0" aria-valuemax="100">50%</div>
</div>
```

Bootstrap modals are dialog boxes or pop-up windows that overlay the current page content. They are commonly used to display additional content, forms, or messages without navigating to a new page. Bootstrap provides a set of classes and JavaScript components to create and manage modals. Here's how to create Bootstrap modals:

Basic Modal Structure:

A Bootstrap modal consists of a trigger button, a modal dialog, and content within the dialog. The modal dialog is initially hidden and becomes visible when triggered.

```
<!-- Trigger button -->
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#myModal">
  Open Modal
```

```
</button>
<!-- Modal -->
<div class="modal" id="myModal">
  <div class="modal-dialog">
    <div class="modal-content">
      <!-- Modal content goes here -->
    </div>
  </div>
</div>
```

Bootstrap provides a powerful and customizable navigation bar component that you can use to create responsive and attractive navigation menus for your website. Here's how to create Bootstrap navigation bars:

Basic Navbar Structure:

A Bootstrap navigation bar typically consists of a <nav> element with the class .navbar, a container for your menu items, and the menu items themselves. Here's a basic structure for a Bootstrap navbar:

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container">
    <a class="navbar-brand" href="#">Your Website</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <ul class="navbar-nav">
        <li class="nav-item active">
          <a class="nav-link" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">About</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Services</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Contact</a>
        </li>
      </ul>
```

```
   </div>
  </div>
</nav>
```

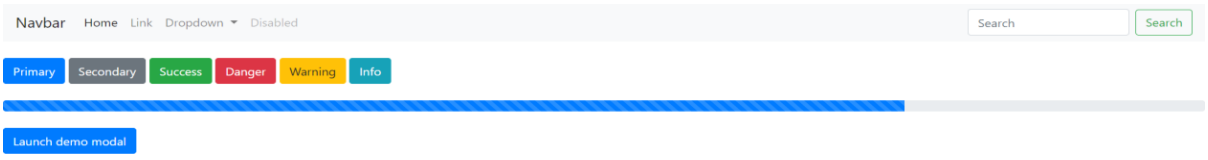
CODE:

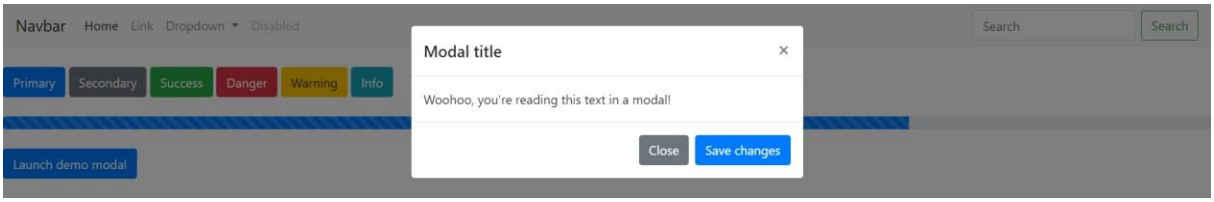
```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>
  <!-- Navigation Bar -->
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false"
aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item active">
          <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Link</a>
        </li>
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-
toggle="dropdown" aria-haspopup="true" aria-expanded="false">
            Dropdown
          </a>
          <div class="dropdown-menu" aria-labelledby="navbarDropdown">
            <a class="dropdown-item" href="#">Action</a>
            <a class="dropdown-item" href="#">Another action</a>
            <div class="dropdown-divider"></div>
            <a class="dropdown-item" href="#">Something else here</a>
```

```
      </div>
     </li>
     <li class="nav-item">
      <a class="nav-link disabled" href="#">Disabled</a>
     </li>
    </ul>
    <form class="form-inline my-2 my-lg-0">
     <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
     <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
    </form>
   </div>
  </nav>
  <br>

  <!-- Buttons -->
  <button type="button" class="btn btn-primary">Primary</button>
  <button type="button" class="btn btn-secondary">Secondary</button>
  <button type="button" class="btn btn-success">Success</button>
  <button type="button" class="btn btn-danger">Danger</button>
  <button type="button" class="btn btn-warning">Warning</button>
  <button type="button" class="btn btn-info">Info</button>
  <br><br>

  <!-- Progress Bar -->
  <div class="progress">
   <div class="progress-bar progress-bar-striped progress-bar-animated" role="progressbar" aria-valuenow="75" aria-valuemin="0" aria-valuemax="100" style="width: 75%"></div>
  </div><br>

  <!-- Modals -->
  <!-- Button trigger modal -->
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal">
   Launch demo modal
  </button><br>

  <!-- Modal -->
  <div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
   <div class="modal-dialog" role="document">
```

```
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">Modal title</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
          Woohoo, you're reading this text in a modal!
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save changes</button>
      </div>
    </div>
  </div>
</div>


 <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
 <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.min.js"></script>
 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

OUTPUT:





Experiment 4: To create and setup the Git repository on Bitbucket or github using SSH.

In the context of GitHub, SSH stands for Secure Shell. SSH is a protocol used for securely connecting to remote servers or services over the internet. GitHub uses SSH to provide secure access to your Git repositories hosted on the platform. Here's how it works:

SSH Key Pair: To interact with your GitHub repositories over SSH, you need to generate an SSH key pair on your local computer. The key pair consists of a public key and a private key. The public key is stored on your GitHub account, while the private key remains on your local machine.

Authentication: When you attempt to perform actions like pushing or pulling code from a GitHub repository, your local Git client uses your private key to authenticate with GitHub. GitHub, in turn, checks your public key to verify your identity.

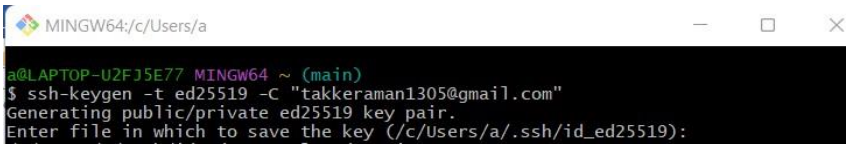
Security: Using SSH keys for authentication is more secure than using a username and password. SSH keys are long, randomly generated character strings, making them difficult for unauthorized users to guess or brute-force.

Ease of Use: Once your SSH key is set up on GitHub, you won't need to enter your username and password each time you interact with your repositories. This makes the process of working with Git and GitHub more convenient.

Setting up SSH for use with Git involves generating an SSH key pair, adding the public key to your GitHub account, and configuring your local Git client to use SSH for authentication. Here's how to do it:

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 generated an SSH key, you can do so by running the following command in your terminal or command prompt:



2. **Copy the SSH Key:**
   After generating the SSH key, you need to copy it to your clipboard. You can do this using the following command:

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

   If you're not on macOS, you can open the public key file in a text editor and manually copy the key.

3. **Add the SSH Key to Your GitHub Account:**

- Open your GitHub account in a web browser and follow these steps:
- Click on your profile picture in the top right and select "Settings."

- In the left sidebar, click on "SSH and GPG keys."
- Click the "New SSH key" button.
- Give your key a title (e.g., "My SSH Key").
- Paste the copied SSH key into the "Key" field.
- Click the "Add SSH key" button.

4. **Create a New Repository:**

- To create a new repository, follow these steps:
- Click the "+" icon in the upper right corner and select "New repository."
- Fill in the repository name and any other desired settings.
- Choose whether to initialize the repository with a README, .gitignore, or license (optional).
- Click the "Create repository" button.

5. **Clone the Repository:**

After creating the repository, you can clone it to your local machine using SSH. In the repository's main page, click the "Code" button, and select "SSH" to get the SSH URL. Then, use the following command to clone the repository:

```
$ git clone git@github.com:yourusername/repository-name.git
```

Replace yourusername with your GitHub username and repository-name with the name of your repository.

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:

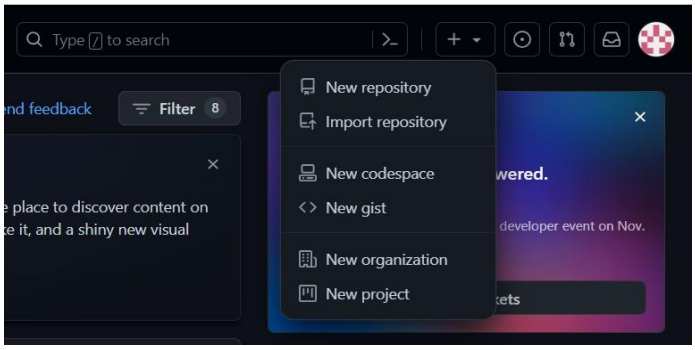
```
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_rsa
```

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

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

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

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
```

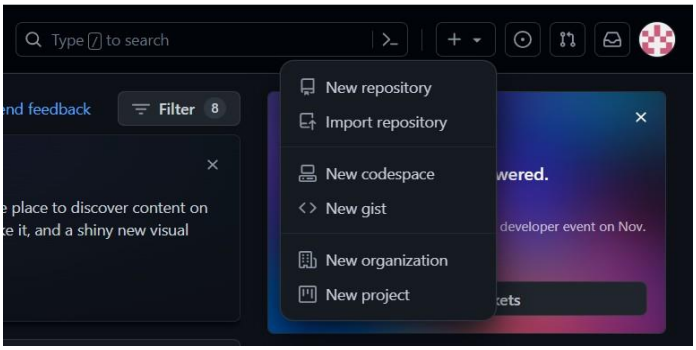
8. Push Your Code to GitHub: Finally, push your code to the GitHub repository using the following command:

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

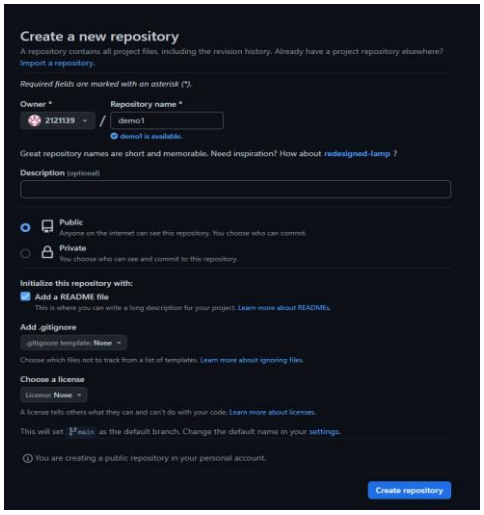
Experiment 5: To perform push, clone and patch operation to Git repository.

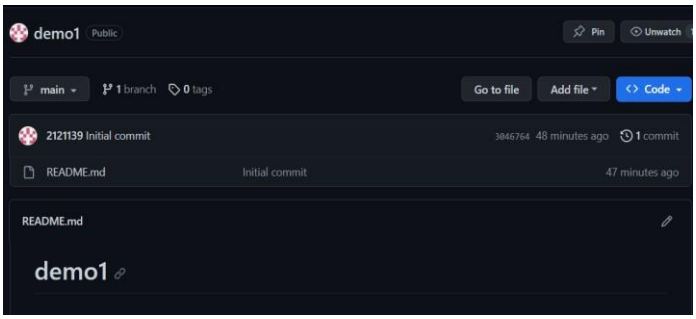
To create a repository:

1. Assuming you already have an account, login to your Github and choose the plus button next to your account picture, click New Repository.



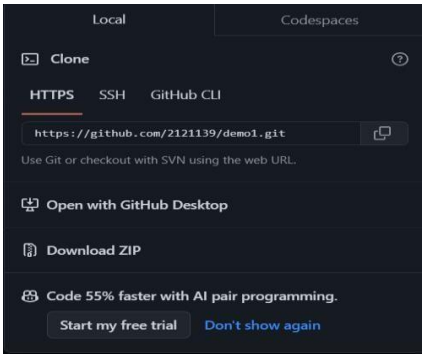
2. Next fill out the name of your repository and initialize it with a README.





To clone a repository:

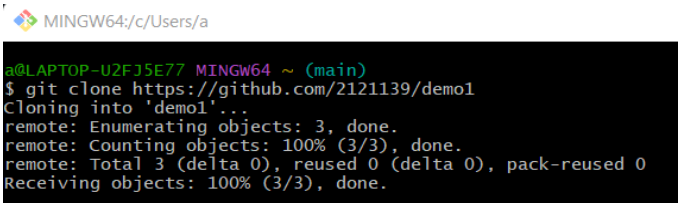
1. Now you'll want to be able to make changes to your Github repository locally. Go to the repository you want to clone on github.com or the one you just created and click the green "Clone or download" button.



2. Copy the link you see in the box and open the Command Prompt. Change into the directory where you want to clone your repository. Use the command below to change directories if you never use terminal/command prompt.

cd foldername

3. Enter this command and paste your link.



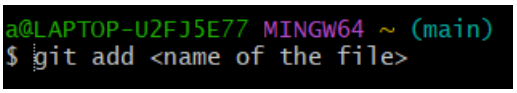
4. Hit enter, the repository should start cloning into your chosen directory.

Push Commits To a Git Repository:

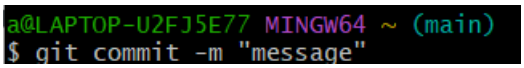
By following the below steps we can push the commit to the git repository.

Step 1: Make sure that your local and Git repositories are up.
Step 2: Stage the modified files using the command line below.

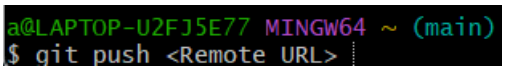
(.) represents all the untracked files. If you want to move a specific file then you can the following command.



Step 3: Commit the staged files into the local repository using the following command. Provide a commit message that details the changes you made and is descriptive.



Step 4: Push the commit to the remote repository from there other developers can access the code. Use the following command.



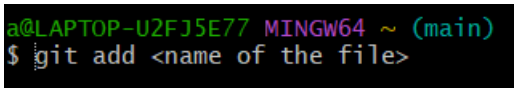
In the event that conflicts arise during the push, resolve them manually before going through steps 2 and 4 again to push the modifications.

Note: You must have the required access and permissions before you can push your commits to a Git repository.
Push Commits To a CodeCommit In Amazon Web Services (AWS)
Follow the steps mentioned below to push the commit to CodeCommit in Amazon Web Services (AWS):
Step 1: Make sure that you have permission to access the CodeCommit in Amazon Web Services(AWS).
Step 2: Use the below command you can clone the repository needed to your local repository.

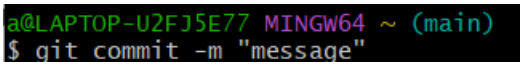
git clone <HTTPS or SSH URL>

Step 3: Stage the modified files using the command below (git add).

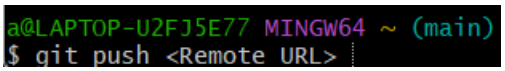
(.) represents all the untracked files. If you want to move a specific file then you can the following command.



Step 4: Commit the staged files into the local repository using the following command. Provide a commit message that details the changes you made and is descriptive.



Step 5: To push changes to code commit us the following command.



The remote URL should be in the format shown below.

https://git-codecommit.[region].amazonaws.com/v1/repos/[repository-name].

Example: To push commits to the CodeCommit repository.

git push <AWS Code Commit URL>

Step 6: Repeat the above 3rd and 5th step if you face any problems like merge conflicts.
If you encounter any conflicts during the push, resolve them manually, and then repeat steps 3 to 5 to push the changes again.

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
```

Experiment 6: To install and setup the CodeIgniter Framework and to understand its MVC architecture.

CodeIgniter Installation

Follow given steps to install CodeIgniter:

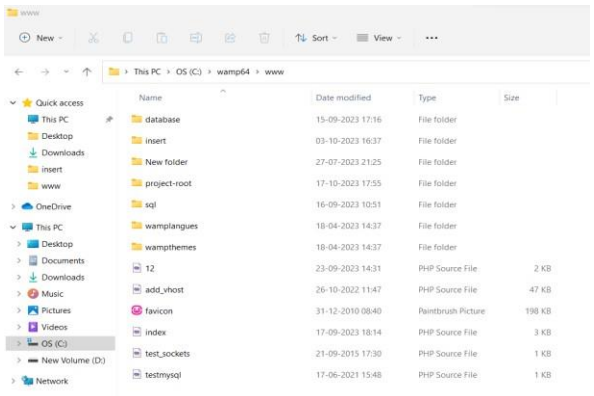
1) Download CodeIgniter from its official website.

Download current version of CodeIgniter from its official website

https://www.codeigniter.com

2) Unzip CodeIgniter package.

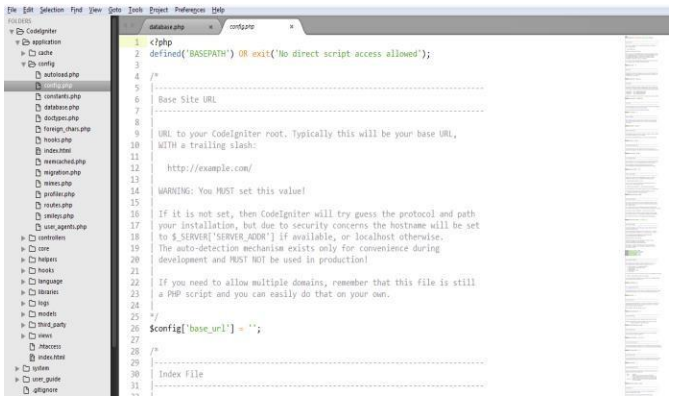
Downloaded CodeIgniter will be in zip format. Copy it and place it in your htdocs folder. Unzip and rename it. We are naming it as **CodeIgniter.**



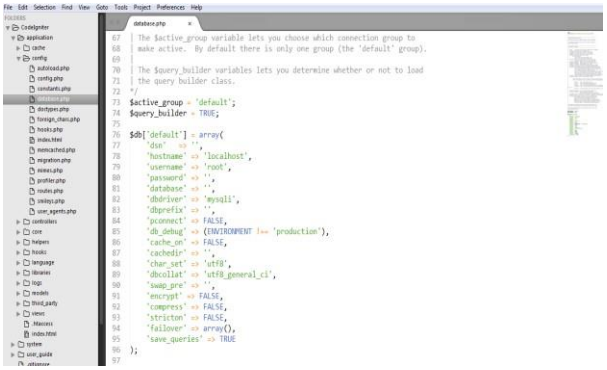
3) CodeIgniter user guide

On browser type **localhost/CodeIgniter/** (after localhost type name of your unzipped folder). If the above snapshot page appears then it means your file is successfully installed.

4) Set the base URL in application/config/config.php file with any text editor.



5) You need to establish the connectivity to your database. Go to the path application/config/database.php file.



Look at the above snapshot, fill the details about your database like hostname, username, password and database name.

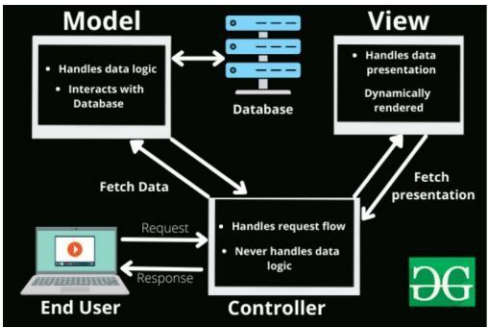
MVC Framework

The Model-View-Controller (MVC) framework is an architectural/design pattern that separates an application into three main logical components Model**,** View**,** and Controller**.** Each architectural component is built to handle specific development aspects of an application. It isolates the business logic and presentation layer from each other. It was traditionally used for desktop graphical user interfaces (GUIs)**.** Nowadays, MVC is one of the most frequently used industry-standard web development frameworks to create scalable and extensible projects. It is also used for designing mobile apps.

Components of MVC :

The MVC framework includes the following 3 components:

- Controller
- Model
- View



Controller:

The controller is the component that enables the interconnection between the views and the model so it acts as an intermediary. The controller doesn't have to worry about handling data logic, it just tells the model what to do. It processes all the business logic and incoming requests, manipulates data using the Model component, and interact with the View to render the final output.

View:

The View component is used for all the UI logic of the application. It generates a user interface for the user. Views are created by the data which is collected by the model component but these data aren't taken directly but through the controller. It only interacts with the controller.

Model:

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. It can add or retrieve data from the database. It responds to the controller's request because the controller can't interact with the database by itself. The model interacts with the database and gives the required data back to the controller.

Advantages of MVC:

- Codes are easy to maintain and they can be extended easily.
- The MVC model component can be tested separately.
- The components of MVC can be developed simultaneously.
- It reduces complexity by dividing an application into three units. Model, view, and controller.
- It supports Test Driven Development (TDD).
- It works well for Web apps that are supported by large teams of web designers and developers.
- This architecture helps to test components independently as all classes and objects are independent of each other
- Search Engine Optimization (SEO) Friendly.

Disadvantages of MVC:

- It is difficult to read, change, test, and reuse this model
- It is not suitable for building small applications.
- The inefficiency of data access in view.
- The framework navigation can be complex as it introduces new layers of abstraction which requires users to adapt to the decomposition criteria of MVC.
- Increased complexity and Inefficiency of data

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

Login is a process where a user logs into your application using a registered email and password. After successfully logging into an application, a user can access the further resources of the application.

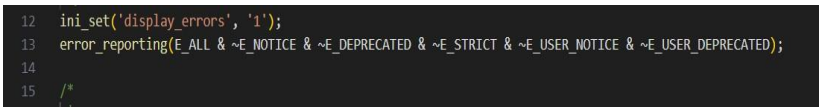
In contrast, signup is when users register themselves using the name, email, and password properties.

Codeigniter 4 Auth (Signin and Signup) System Example:

Step 1: Create Codeigniter Project
Step 2: Display Errors
Step 3: Generate Table into Database
Step 4: Connect CI to Database
Step 5: Create and Update User Model
Step 6: Register Auth Controllers
Step 7: Create Auth View
Step 8: Protect Route with Filter
Step 9: Run CI Application

Display Errors

You may turn on the feature to errors, go to the **app/Config/Boot/production.php** and change display_errors prop value to 1 from 0.



Connect CI to Database

The existing step, describes how you can connect CI app to database, its amazingly facile process, add database name, username and password in **app/Config/Database.php.**

```
0 references
public array $default = [
    'DSN'          => '',
    'hostname'     => 'localhost',
    'username'     => 'root',
    'password'     => '',
    'database'     => 'codeigniter_db',
    '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,
];
```

Create and Update User Model

Further, create a new model file, and define the users table name and the values in the $table and $allowedFields. Create UserModel.php file in app/Models folder after that update the given code in **app/Models/UserModel.php** file.

```
app > Models > UserModel.php > PHP Intelephense > UserModel
1   <?php
2   namespace App\Models;
3   use CodeIgniter\Model;
4
    4 references | 0 implementations
5   class UserModel extends Model{
        36 references
6       protected $table = 'users';
7
        14 references
8       protected $allowedFields = [
9           'name',
10          'email',
11          'password',
12          'created_at'
13      ];
14  }
```

Register Auth Controllers

A controller is a file that holds the functions and methods used to handle the application's business logic; in this step, you have to create Home.php file in controller in the app/Controllers directory, then insert the below code intothe **app/Controllers/Home.php** file.

Contains Home class by extending basecontroller in which insert delete edit and update functions are being made containing different functionalities

```php
<?php

namespace App\Controllers;
use App\Models\UserModel;

class Home extends BaseController
{
    public function index(): string
    {
        return view('welcome_message');
    }
    public function insert(){
        $data = [
            'user_name'=>$this->request->getVar('user_name'),
            'user_email'=>$this->request->getVar('user_email'),
            'user_password'=>$this->request->getVar('user_password'),
        ];

    $model=new UserModel();

    $model->insert($data);
    echo "<h1>Data Sent Successfully...</h1>";
    }
```

```php
public function show(){
    $model = new UserModel();

    $data['users'] = $model->findAll();

    return view('Show', $data);
}
```

```php
public function delete($id=null){
    $model = new UserModel();

    $data['users'] = $model->where('user_id', $id)->delete();

    return redirect()->to(base_url('Home/show'));
}
```

```php
public function edit($id=null){
    $model = new UserModel();
    $data['users'] = $model->where('user_id', $id)->first();

    return view('edit', $data);
}

//UPDATED data store  in array
public function update(){
    $data = [
        'user_name'=>$this->request->getVar('user_name'),
        'user_email'=>$this->request->getVar('user_email'),
        'user_password'=>$this->request->getVar('user_password'),
    ];
    // getVar to get value of fields
    $id = $this->request->getVar('user_id');
    $model = new UserModel();
   // on base of id we will update record
   //data = to be updated
    $model->update($id, $data);
    return redirect()->to(site_url('Home/show'));

    //redirect to given location and show() called to get data


}
```

Create Auth View Templates

So far, we have followed every instruction to proliferate this Codeigniter auth system example, and now we have to define the view files.

Make sure to get inside the app/Views folder and create signin.php and signup.php files; these files will be used for login and user registration in Codeigniter.

Create the user registration form, Open to access data **app/View/welcome_message.php** file.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Registeration</h1>
    <form method="POST" action="<?php echo site_url('Home/insert'); ?>">
    <label><b>User Name:</b></label>
    <input type="text" name="user_name"><br><br>
    <label><b>User Email:</b></label>
    <input type="text" name="user_email"><br><br>
    <label><b>User Password:</b></label>
    <input type="password" name="user_password"><br><br>
    <input type="submit" name="submit" value="Submit">
</form>
</body>
</html>
```

For editing data
app/View/edit.php file.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Update Registeration</h1>
    <form method="POST" action="<?php echo site_url('Home/update'); ?>">
    <input type="hidden" name="id" value="<?php echo $users['user_id'];?>"><br><br>
    <label><b>User Name:</b></label>
    <input type="text" name="user_name" value="<?php echo $users['user_name'];?>"><br><br>
    <label><b>User Email:</b></label>
    <input type="text" name="user_email" value="<?php echo $users['user_email'];?>"><br><br>
    <label><b>User Password:</b></label>
    <input type="password" name="user_password" value="<?php echo $users['user_password'];?>"><br><br>
    <input type="submit" name="submit" value="Update">
</form>
</body>
</html>
```

Lastly, move towards routes file to create routes, we need to set routes to execute the controller functions. Similarly, protect the profile route, which will be restricted for un-authenticated users.Get into the **app/Config/Routes.php** file and define the given routes into the file.

```php
<?php

use CodeIgniter\Router\RouteCollection;

/**
 * @var RouteCollection $routes
 */
$routes->get('/', 'Home::index');
$routes->post('Home/insert', 'Home::insert');
$routes->get('Home/show', 'Home::show');
$routes->get('Home/delete/(:num)', 'Home::delete/$1');
$routes->get('Home/edit/(:num)', 'Home::edit/$1');
$routes->post('Home/update', 'Home::update');
//$1 used to pass user id
```

Run CI Application

Eventually, now you have landed on the last section of this tutorial, and we will advise you to use the given command to run the CI app.

php spark serve

You are ready to signup in Codeigniter, go ahead and use the provided url.

http://localhost:8080/signup

Registeration

User Name:

User Email:

User Password:

Submit

User Name: mitchel

User Email: mitchel88@gmail.com

User Password: ••••••

Update

CURD Operations:

View:

User Id	User Name	User Email	User Password
1	santa singh	santasingh9463@gmail.com	sant456
2	harjot singh	harjot745@gmail.com	har45
3	shubham	shubh780@gmail.com	shubh55
4	freakin	freakin90@gmail.com	frek123
5	mitchel	mitchel88@gmail.com	mit234
6	torat	torat@gmail.com	tor35

☐	Edit	Copy	Delete	1	santa singh	santasingh9463@gmail.com	sant456
☐	Edit	Copy	Delete	2	harjot singh	harjot745@gmail.com	har45
☐	Edit	Copy	Delete	3	shubham	shubh780@gmail.com	shubh55
☐	Edit	Copy	Delete	4	freakin	freakin90@gmail.com	frek123
☐	Edit	Copy	Delete	5	mitchel	mitchel88@gmail.com	mit234
☐	Edit	Copy	Delete	6	torat	torat@gmail.com	tor35

Delete User:

User Id	User Name	User Email	User Password	Delete
1	santa singh	santasingh9463@gmail.com	sant456	Delete
2	harjot singh	harjot745@gmail.com	har45	Delete
3	shubham	shubh780@gmail.com	shubh55	Delete
4	freakin	freakin90@gmail.com	frek123	Delete
5	mitchel	mitchel88@gmail.com	mit234	Delete
6	torat	torat@gmail.com	tor35	Delete

☐	Edit	Copy	Delete	1	santa singh	santasingh9463@gmail.com	sant456
☐	Edit	Copy	Delete	2	harjot singh	harjot745@gmail.com	har45
☐	Edit	Copy	Delete	3	shubham	shubh780@gmail.com	shubh55
☐	Edit	Copy	Delete	4	freakin	freakin90@gmail.com	frek123
☐	Edit	Copy	Delete	5	mitchel	mitchel88@gmail.com	mit234

After Edit:

User Id	User Name	User Email	User Password	Delete	
1	santa singh	santasingh9463@gmail.com	sant456	Delete	Edit
2	harjot singh	harjot745@gmail.com	har45	Delete	Edit
3	shubham	shubh780@gmail.com	shubh55	Delete	Edit
4	freakin	freakin90@gmail.com	frek123	Delete	Edit
5	mitti	mitti88@gmail.com	mit234	Delete	Edit

Update Registeration

User Name: mitchel

User Email: mitchel88@gmail.com

User Password: ••••••

Update

User Id	User Name	User Email	User Password	Delete	
1	santa singh	santasingh9463@gmail.com	sant456	Delete	Edit
2	harjot singh	harjot745@gmail.com	har45	Delete	Edit
3	shubham	shubh780@gmail.com	shubh55	Delete	Edit
4	freakin	freakin90@gmail.com	frek123	Delete	Edit
5	mitchel	mitchel88@gmail.com	mit234	Delete	Edit

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

Laravel | Installation and Configuration

Laravel is a PHP framework that makes building complex web applications a cakewalk. In this guide you are going to learn how to Install and Configure Laravel and get your first Laravel project running in your computer.

Prerequisites:

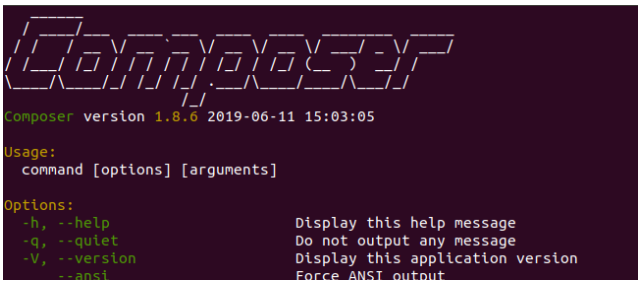
- Basic knowledge of PHP
- PHP Installation
- Basic knowledge of Terminal/Command Prompt

Steps to install Laravel:

- Laravel actually uses a dependency manager tool called Composer to install all required libraries to run Laravel. So before we move forward make sure that composer is installed in your system. It is well documented and after installation run the below given command in your terminal/command prompt:

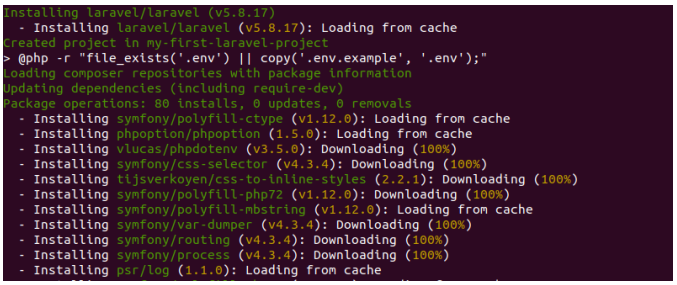
composer

If it gives an output like given below then it means the installation was successful and we can go to the next step.



- Next step is to create a folder where we are going to create our new Laravel Project. After moving to that folder just run this composer command to create a new Laravel Project. composer create-project --prefer-dist laravel/laravel my-first-laravel-project

Here, create-project is a composer command which is just downloading the package and installing all the dependencies further needed by this package and then we have this –prefer-dist which means just prefer the stable version releases if possible, next we mention name of package which is laravel/laravel and my-first-laravel-project is actually the name of project folder and you can replace it with any name that you want. This will start the installation of all dependencies as given in the image below.



If everything went well then end of this script will look like this:

```
filp/whoops suggests installing whoops/soap (Formats errors as SOAP responses)
sebastian/global-state suggests installing ext-uopz (*)
phpunit/php-code-coverage suggests installing ext-xdebug (^2.6.0)
phpunit/phpunit suggests installing phpunit/php-invoker (^2.0)
phpunit/phpunit suggests installing ext-soap (*)
phpunit/phpunit suggests installing ext-xdebug (*)
Writing lock file
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: beyondcode/laravel-dump-server
Discovered Package: fideloper/proxy
Discovered Package: laravel/tinker
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Package manifest generated successfully.
> @php artisan key:generate --ansi
Application key set successfully.
```

- Now you have a new folder called my-first-laravel-project and move to that folder. You will notice that inside this folder a lot of files and folders are already present, this is actually the Laravel framework itself. In this folder, run the command given below to start the in-build PHP server.

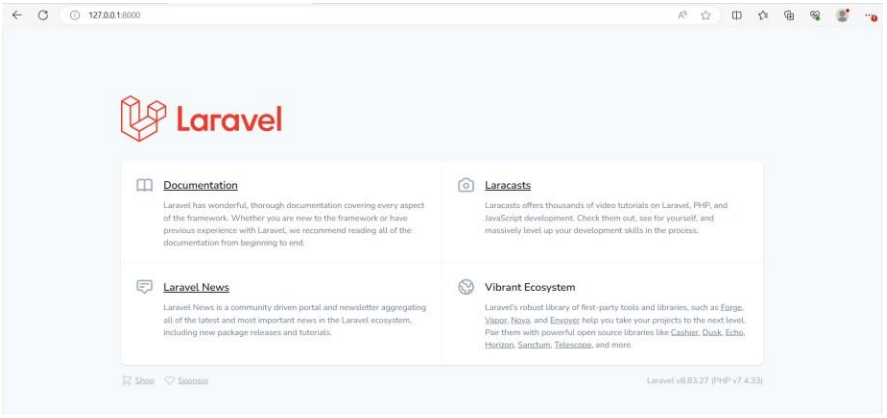
```
C:\wamp64\www\demo>php artisan serve
```

This will give an output like given below:

```
Laravel development server started: <http://127.0.0.1:8000>
```

The link really depends on your system, it can be http://localhost:8000 also.

- Copy this link and open it in your browser to see your Laravel app (It is **http://127.0.0.1:8000**). It will look like this:



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

Step 1: Install Laravel

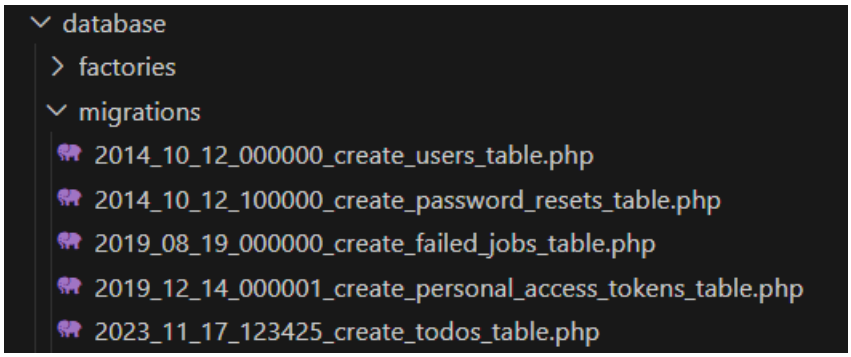
Step 2: Update DB Credentials

Now go the .env and change the DB_DATABASE, DB_USERNAME, DB_PORT and DB_PASSWORD according to your database details.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=todo
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DRIVER=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120
```

Step 3: Migrate Database

Laravel migration is one of the laravel's popular feature which let us create, modify and delete tables and its columns.Laravel ships some of the migration files for some tables by default which includes users, password resets, failed_jobs and personal_access_tokens and migrations tables. And you can see them by going to database/migrations folder:




```
PS C:\wamp64\www\todos> php artisan migrate
```

And if you open the database you will see that your database will now have 6 tables which I have mentioned above:

	Table	Action	Rows	Type	Collation	Size	Overhead
☐	failed_jobs	Browse Structure Search Insert Empty Drop	0	MyISAM	utf8mb4_unicode_ci	4.0 KiB	-
☐	migrations	Browse Structure Search Insert Empty Drop	6	MyISAM	utf8mb4_unicode_ci	2.3 KiB	-
☐	password_resets	Browse Structure Search Insert Empty Drop	0	MyISAM	utf8mb4_unicode_ci	4.0 KiB	-
☐	personal_access_tokens	Browse Structure Search Insert Empty Drop	0	MyISAM	utf8mb4_unicode_ci	4.0 KiB	-
☐	todos	Browse Structure Search Insert Empty Drop	3	MyISAM	utf8mb4_unicode_ci	2.3 KiB	92 B
☐	users	Browse Structure Search Insert Empty Drop	0	MyISAM	utf8mb4_unicode_ci	4.0 KiB	-

Step 4: Run Application

Now in order to start application, you need to type following command in the terminal:

```
C:\wamp64\www\todos>php artisan serve
```

Step 5: Create Migration for the todos table

Since we are creating the todo app, we need to create the todo table, in which we will store all todo.

We will have the following columns in todo table: title_is_completed

Let's create the migration file for the todo table by typing the following command:

```
php artisan make:migration create_todos_table
```

This will create a new migration file in the database/migrations folder with the name of create_todos_table.php with some date prefix which will contains some following code:

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

0 references | 0 implementations
class CreateTodosTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    0 references | 0 overrides
    public function up()
    {
        Schema::create('todos', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('work');
            $table->string('dueDate');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    0 references | 0 overrides
    public function down()
    {
        Schema::dropIfExists('todos');
    }
}
```

Step 6: Create Model for the todos table

```
php artisan make:model Todo
```

Setting up the model:

If you open that up, you will see something like this:

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

6 references | 0 implementations
class todo extends Model
{
    use HasFactory;
    0 references
    protected $table ='todos';
    0 references
    protected $primary_key='id';
}
```

Step 7: Create Controller and Route

```
php artisan make:Controller TodoController --resource
```

This will allow us to create the controller, where as --resource is to declare as the resource which create some empty functions like index, create, store, show, edit, update anddestroy to save time that we will use these functions to add our business logic to make CRUD operation.

Route is a way of requesting data from end users. The route is usually tied with controller function.

In simple terms it means that if the user types this in URL execute that controller funtion which is tied to that route.

```
Route::resource('todos',
\App\Http\Controllers\TodoController::class);
```

Step 8: Create View File

```
<div class="container">
    <div class="d-flex justify-content-between align-items-center my-5">
        <div class="h2">
            Update Todo
        </div>
        <a href="" class="btn btn-primary btn-lg">Back</a>
    </div>


    <div class="card">
        <div class="card-body">
            <form action="{{route("todo.updateData")}}" method="post">
                @csrf
                <label for="" class="form-label mt-4">Name</label>
                <input type="text" name="name" class="form-control" value="{{$todo->name}}">

                <label for="" class="form-label mt-4">Work</label>
                <input type="text" name="work" class="form-control" value="{{$todo->work}}">

                <label for="" class="form-label mt-4">Due Date</label>
                <input type="date" name="dueDate" class="form-control" value="{{$todo->dueDate}}">
                <input type="number" name="id" value="{{$todo->id}}" hidden>

                <button class="btn btn-primary btn-lg mt-5">Update Todo</button>
            </form>
        </div>
    </div>
```

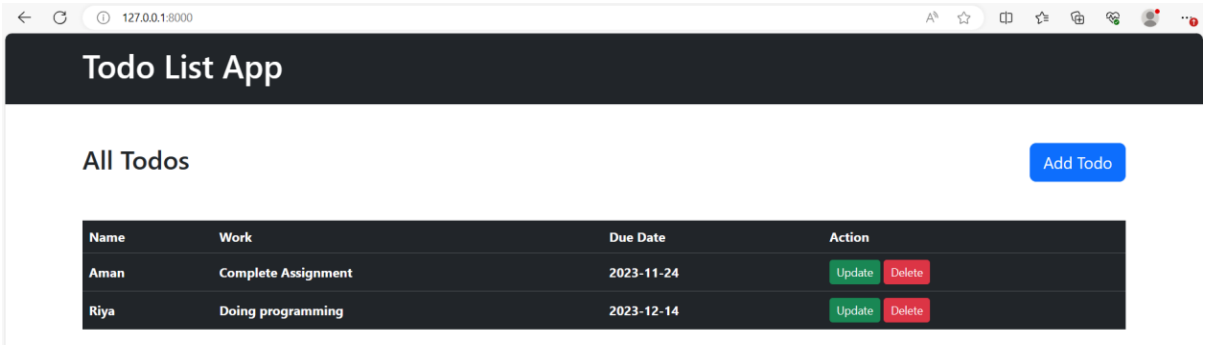
```blade
<div class="card">
    <div class="card-body">
        <form action="{{route("todo.store")}}" method="post">
            @csrf
            <label for="" class="form-label mt-4">Name</label>
            <input type="text" name="name" class="form-control">
            <div class="text-danger">
                @error('name')
                {{$message}}
                @enderror
            </div>

            <label for="" class="form-label mt-4">Work</label>
            <input type="text" name="work" class="form-control">
            <div class="text-danger">
                @error('work')
                {{$message}}
                @enderror
            </div>

            <label for="" class="form-label mt-4">Due Date</label>
            <input type="date" name="dueDate" class="form-control">
            <div class="text-danger">
                @error('dueDate')
                {{$message}}
                @enderror
            </div>

            <button class="btn btn-primary btn-lg mt-5">Add Todo</button>
        </form>
    </div>
</div>
```

Step 9: Run Application

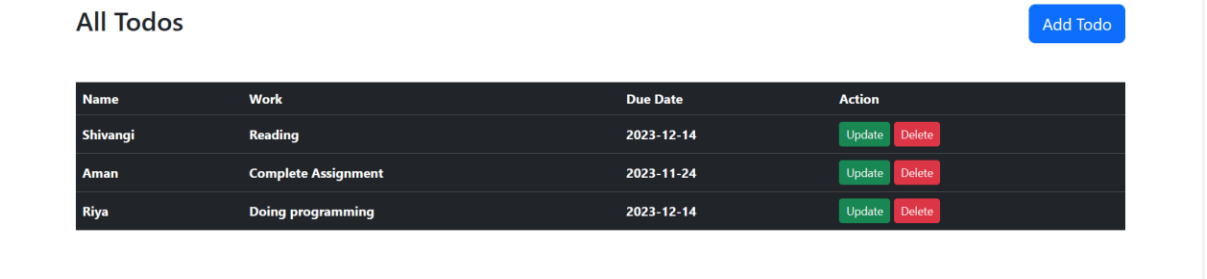
Now in order to start application, you need to type following command in the terminal:

```
C:\wamp64\www\todos>php artisan serve
```

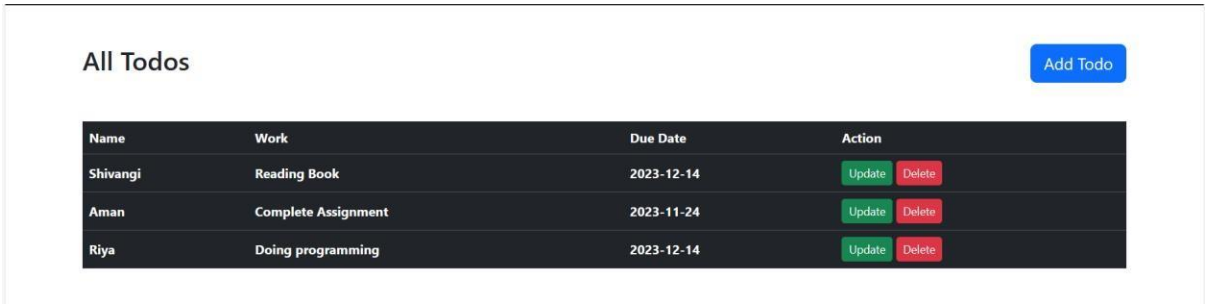
```
Starting Laravel development server: http://127.0.0.1:8000
```



Add Todo:



Update Todo:



Delete an Item: