Name : Aditya Anand Wadkar
Class : TE Roll No: 33379 Div :- I I
Sub : WADL

Assignment I - a

Title :- Web page Design

Aim :- Create a responsive webpage which shows ecommerce / college dashboard with sidebar & statistics in card using HTML, CSS & Bootstrap.

Objective :- Apply HTML, CSS & Bootstrap classes & demonstrate webpage is responsive

Requirement :- Linux OS : Ubuntu, Vscode Editor. PC with configuration as pentium IV

Theory :-

1) **HTML :-**
Hyper text markup language (HTML) is rendered on web Browser. HTML has a set of elements that describes structure.

a) **HTML elements**
i) The HTML element comprises of start tag, content and end tag.
<tagname> content </tagname>
ii) The main HTML element is <html> ... </html>
All HTML will be enclosed within it

b) HTML Boilerplate :

```
<!Doctype>
<html lang = "en">
<head>
    <link rel = "stylesheet" href = "style.css">
</head>
<body>
    One HTML element is
</body>
</html>
```

c) HTML Attributes :-

An attribute is a key-value pair written within an element.

Example :- <a>

i) href : used to mention link
ii) target : indicates where to open document
iii) style : used to add styles

Application of HTML

1) Web page development - HTML is used to create pages that're rendered over web.
2) Responsive UI - HTML provides tags that're used to navigate from one page to another.
3) Offline support - HTML pages once loaded can be made available offline.

Advantages of HTML:-

1) HTML helps to build structure of website & is widely used markup language.
2) Every browser support HTML language
3) HTML is light weighted & fast to load.

Disadvantage of HTML:

1) It cannot produce dynamic output alone
2) Making structure of HTML document becomes tough to understand.
3) Errors can be costly

CSS (cascading style sheet)

CSS is used to style or achieve desired look and feel of web page. CSS has a wide range of properties that can facilitate user from changing color text to applying animations.

a. CSS syntax:

selector { property : value; }

selector could be element or id selector or class seletor etc.

b. Types of css

- Inline css - It is written within HTML element itself using style attribute.

eg

- Internal css - It is written within HTML document using <style> tag.

- External css - It is written in separate css file. css file is saved as with .css extension.

C. css selector

- css selector are various ways of writing css

1. Element selector:
   The element selector uses name of HTML tag
2. ID selector :-
   The id selector uses unique id of HTML tag to apply css
3. Class selector:
   The class selector uses class of HTML tag to apply css
4. universal selector
   The universal selector applies css to each HTML element of web page.

There're four different combination selectors in css

- descendant selector (space)
- child selector (>)
- adjacent sibling selector (+)
- general sibling selector (~)

Application of css:

1) Faster page loading
2) Accelerated Development
3) Improved user Experience
4) Device compatibility
5) Simplified layout changes.

Advantages of css

1) web designer needs to use few lines.
2) less complex

Dis advantages of css

1) multiple version creates confusion in browser.
2) There exist a scarcity
3) Browser compatibility
4) There might be cross-browser issues while using css.

Bootstrap:

- Bootstrap is a light-weight library of CSS
- Bootstrap 5 is common used version of now.
- It has a wide range of class that works perfectly on all browser.

Bootstrap Grid System:-

- Bootstrap grid allows 12 columns in a row in web page
- The Bootstrap 5 grid system has 6 classes
  - .col - (extra small devices)
  - .col-sm - (small devices)
  - .col-md - (medium devices)
  - .col-lg - (large devices)
  - .col-xl - (xlarge devices)
  - .col-xxl - (xxlarge devices)

How to add Bootstrap CDN link to your code

```
<head>
    <link href="link" rel="stylesheet">
    <script src="link"></script> </head>
```

Application of Bootstrap:

1) Bootstrap can be used to create consistent, well-designed web pages & user-interface
2) Bootstrap provides wide range of UI components & features, such as navigation bars, forms, buttons etc.

Advantages of Bootstrap:-

1) Save time
2) well-documented and widely used.

Disadvantage of Bootstrap

1) Because Bootstrap is so widely used, some websit may end up looking very similar to each other and lack a distinctive visual identity

2) Bootstrap can be bloated & slow-down page load times if developers include unnecessary components.

Conclusion :

Thus, we have applied HTML, CSS and Bootstrap classes and demonstrated a web page that is responsive.

16/2

ASSIGNMENT 1a

Name: - Aditya Wadkar

Roll No: - 33379

Class: - TE-11

subject: - WADL

Source code

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <title>Dashboard</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1">
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet">
 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>
 <link rel="stylesheet" href="style.css">
 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
</head>
<body>
  <div class="container-fluid">
    <div class="row flex-nowrap">
      <div class=" d-none d-md-block col-auto col-md-2 col-xl-2 px-sm-2 col-sm-3 px-0 bg-dark">
        <div class="d-flex flex-column align-items-center align-items-sm-start px-3 pt-2 text-white min-vh-100">
          <div class="avatar">
            <img src="https://img.freepik.com/free-vector/businessman-character-avatar-isolated_24877-
60111.jpg?w=740&t=st=1676783616~exp=1676784216~hmac=c193eacc65786ad5dc715c8f271c52371cd32da2a7cca8ee5615759
82b597f98" alt="distressed Italian wall in pink and grey hues">
          </div>
          <div class="center">
            <p> User<br>user@gmail.com</p>
            <hr>
          </div>
            <div class="side-contents">
            <i class="fa fa-sliders" aria-hidden="true" style="margin-top: 0.35rem;"></i>
            <p>DASHBOARD</p>
          </div>
          <div class="side-contents">
            <i class="fa fa-user-circle-o" aria-hidden="true" style="margin-top: 0.35rem;"></i>
            <p >ACCOUNT</p>
          </div>
          <div class="side-contents">
            <i class="fa fa-rss" aria-hidden="true"  style="margin-top: 0.35rem;"></i>
            <p>EXERCISE</p>
          </div>
          <div class="side-contents">
            <i class="fa fa-asterisk" aria-hidden="true" style="margin-top: 0.35rem;"></i>
            <p>DIET</p>
          </div>
          <div class="side-contents">
            <i class="fa fa-user" aria-hidden="true" style="margin-top: 0.35rem;"></i>
            <p>SUBSCRIBE</p>
          </div>
          <div class="side-contents">
```

```
  <i class="fa fa-server" aria-hidden="true" style="margin-top: 0.35rem;"></i>
  <p>SERVICES</p>
</div>
<div class="logout">
  <button type="button" class="btn btn-primary"><i class="fa fa-sign-out" aria-hidden="true"></i> Logout</button>
</div></div></div>
<div class="col py-3">
  <nav class="navbar bg-light">
    <div class="container-fluid">
    <div class ="d-block d-md-none">
      <i class="fa fa-bars" aria-hidden="true" ></i>
    </div>
     <span class="navbar-brand mb-0 h1">Personal Progress</span>
     <button type="button" class="btn btn-dark"> Contact Support</button>
    </div>
  </nav>
  <br><br>
  <div class="row">
    <div class="col-lg-4 col-md-12 col-sm-12 col-xs-1  p-3">
      <div class="card" style="width: 100%;">
        <div class="card-body cb1">
          <div class="card-body cb1">
            <h5 class="card-title">Stats</h5>
            <img src="stats.png" style="width:95%;">
          </div></div></div></div>
      <div class="col-lg-4 col-md-12 col-sm-12 col-xs-1  p-3">
        <div class="card" style="width: 100%;">
          <div class="card-body cb1">
            <div class="card-body cb1">
              <h5 class="card-title">cardio</h5>
              <img src="cardio2.png" style="width:55%;">
            </div></div></div></div>
      <div class="col-lg-4 col-md-12 col-sm-12 col-xs-1  p-3">
        <div class="card" style="width: 100%;">
          <div class="card-body cb1">
            <h5 class="card-title">climbing Inclination</h5>
            <img src="cardio.png" style="width:100%;">
          </div></div></div></div>
    <br><br>
    <div class="row">
      <div class="col-lg-8 col-md-12 col-sm-12 col-xs-1  p-3">
        <div class="card" style="width: 100%;">
          <div class="card-body cb1">
            <h5 class="card-title">Progress Tracking</h5>
            <img src="progress.png" >
          </div></div></div>
      <div class="col-lg-4 col-md-12 col-sm-12 col-xs-1  p-3">
        <div class="card" style="width: 100%;">
          <div class="card-body cb1">
            <h5 class="card-title">weekly Goal Achieved</h5>
            <!-- <div class="card_class">
              <i class="fa fa-file-text-o" aria-hidden="true" style="color: coral"></i>
            </div>
            <br>
            <p class="card-text">Some quick example text to build on the card title and make up the bulk
```

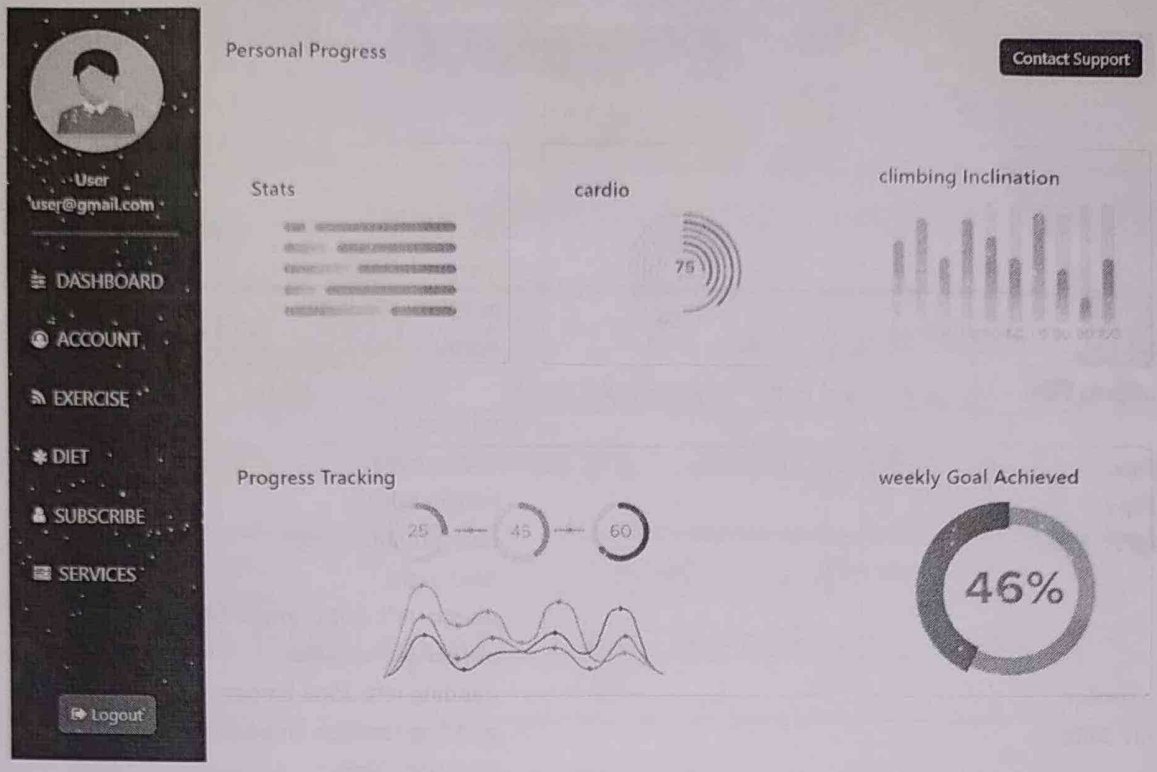
```
                of the card's content.</p> -->
                <img src="goal.png" style="width: 72%;">
            </div> </div></div></div></div>
    </body>
</html>
```

Style.css

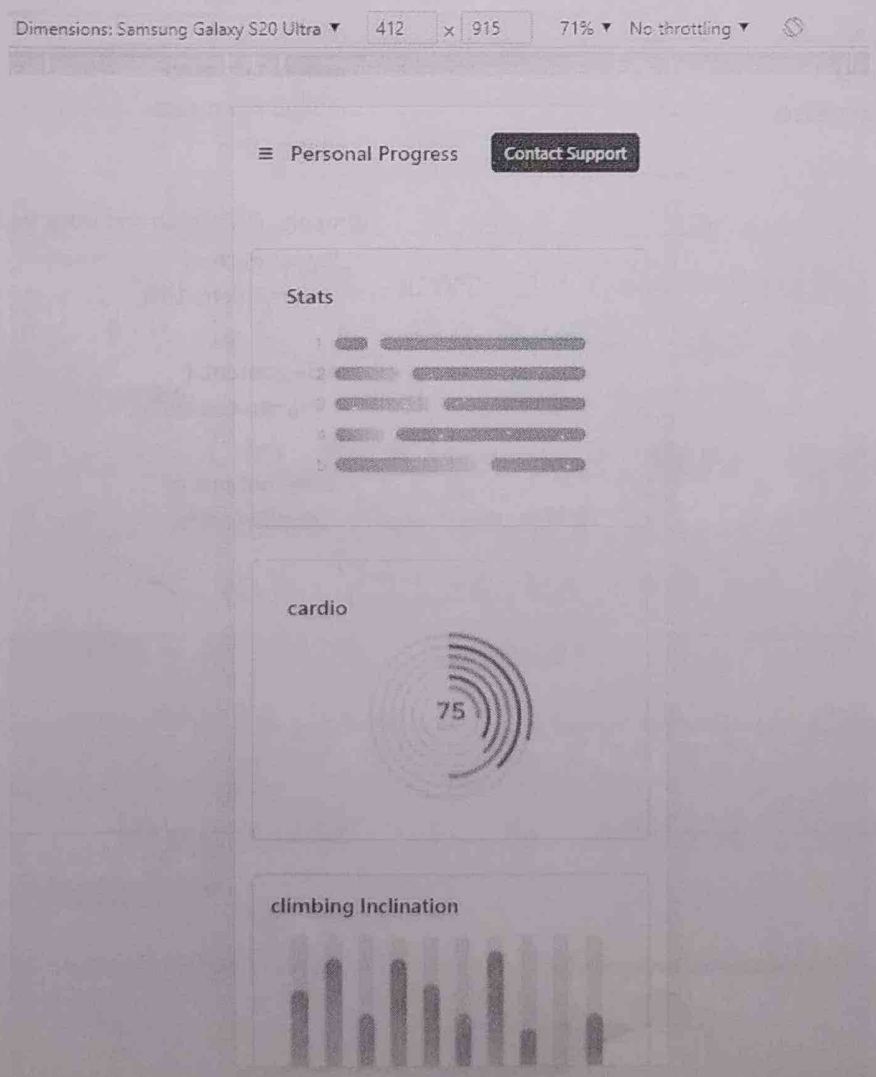
```
*{
    margin: 0;
}
.avatar img {
    margin-top: 12%;
    margin-bottom: 12%;

    width: 130px;
    height: 120px;
    border-radius: 50%;

}
.center p{
    text-align: center;
    font-weight: 500;
}

.center hr{
    /* text-align: center; */
    /* font-weight: 1000; */
    background-color:chartreuse;
    height: 3.5px;
    width: 120%;

}

.side-contents{
    margin-top: 10%;
    display:flex;

}
.side-contents p{
    margin-left: 10%;
    font-size: large;
}
.logout button{
    margin-left:2%;
    position: absolute;
    bottom: 4%;
}
.card_class {
    width: 60px;
    height: 60px;
    background: orange;
    float: right;
    margin-left: 10px !important;
    border-radius: 50%;
    padding-left: 15px !important;
    padding-top: 8px !important;
    font-size: 30px;
}
.card-body img{
    display: block;
    margin-left: auto;
    margin-right: auto;
    width: 50%;
}
@media only screen and (max-width: 600px) {
    .logout button{
        margin-left:-10%;
    }
    .side-contents{
        margin-top: 20%;
    }
    .side-contents p{
        display: none;
    }
}
```

Output:

Desktop View:



Mobile View:



Name: Aditya wadkar
class TE I1 batch: N I1
Roll no: 33379

PICT, PUNE

Assignment 1b

Title : Javascript Ajax implementation

Problem statement: write a javascript program to get user registration data and push to array /local storage with AJAX

Objective : To understand working of JS, AJAX xmlHttpRequest by accepting user registration data.

Requirements : Ubuntu/Windows, Vscode.

Theory :-

1) **HTML form**:
An HTML form is collection of textbook
- To fill info like name roll no etc.
- Textarea to fill info Address
- select (to select value from dropdown list)
- Radio button to make one selection line
- check box to select one or more option from limited choice.

2) **Javascript**:
Javascript is a frontend scripting language.

Javascript can be added to HTML file in :

1] Internal Javascript :

Here, the code is directly added to HTML file using <script></script> tags.

2] External Javascript :

A seperate file with .js extension is prepared where only Javascript code is written.

3. XML Http Request :

XMLHttp Request object is an API used to fetch data from ~~from~~ server XMLHTTP. It can retrive any format of data such as Json XML text etc.

Properties :

- onload : when request object is sensed & response is ready it define a fuction to called
- onreadystatechange : A function is called whenever ready state property changes
- ready state : It hold current status of XML There're 5 states of request.
  - ready state = 0 : it represent request not initialized
  - ready state = 1 : Establishment of server connection
  - ready state = 2 : Request has been Received
  - ready state = 3 : During time of processing request

- ready_state = 4 : Response is ready after finishing request.

- Response Text : It returns response data received by request in form of string
- Response XML : It returns responsedata by request in xml format.

---

4. AJAX :

Ajax is abbreviation for Asynchronous JS. Ajax also hits requests to server & loads the data on HTML without page request.
applications :- Google maps, youtube.

with jQuery AJAX method, one can request text, HTML, XML or JSON from sensor using HTTP Get / or HTTP POST.

The parameter specify one or more value pairs for AJax request

Name	value / Description
async	A boolean value indicating whether request should be handled asynchronous or not default
content type	The content type used when sending data to server.

data	specifies data to be sent to server
datatype	The datatype expected of the senor response.
error (xhr, status)	A function to run if request fails.
success (result)	A function to run if request succeeds.
type	specific the type of request (GET or POST)
URI	specifices url to send request to default is content page.

Conclusion:

Thus we have implemented a Javascript program to get user registration data and using AJax post method we have listed on new page.

ASSIGNMENT 1b

Name: - Aditya Wadkar

Roll No: - 33379

Class: - TE-11

subject: - WADL

Source code

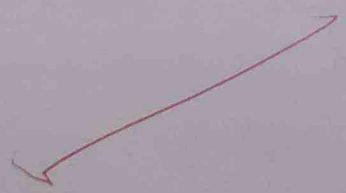
Index.html

```
<!doctype html>
<html lang="en">
<head>
	<title>Assignment 1B</title>
	<meta charset="utf-8">
	<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
	<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-iYQeCzEYFbKjA/T2uDLTpkwGzCiq6soy8tYaI1GyVh/UjpbCx/TYkiZhlZB6+fzT"
crossorigin="anonymous">
	<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.3/jquery.min.js"></script>
</head>
<body>
	<div class="container my-4">
		<form id="addNumberForm" class="p-4 border shadow-sm w-50 mx-auto">
			<div class="mb-2">
				<label for="firstName" class="form-label">Name</label>
				<input  type="text" class="form-control" name="firstName" required id="firstName"
aria-describedby="helpId" placeholder="Enter Your Name">
			</div>
			<div class="mb-2">
				<label for="Email" class="form-label">Email Address </label>
				<input type="email" class="form-control" name="Email" required id="Email" aria-
describedby="helpId" placeholder="Enter Your Email Address">
			</div>
			<div class="mb-2">
				<label for="age" class="form-label">Age</label>
				<input  type="text" class="form-control" name="age" required id="age" aria-
describedby="helpId" placeholder="Enter Your Age">
			</div>
			<div class="mb-3">
				<label for="college" class="form-label">College Name</label>
				<input type="text" class="form-control" name="college" required id="college" aria-
describedby="helpId" placeholder="Enter Your College Name">
			</div>
			<button type="submit" class="btn btn-primary">Submit</button>
			<hr>
			<small class="small text-success" id="msg"></small>
		</form>
	</div>

	<script>
		function fetchData() {
```

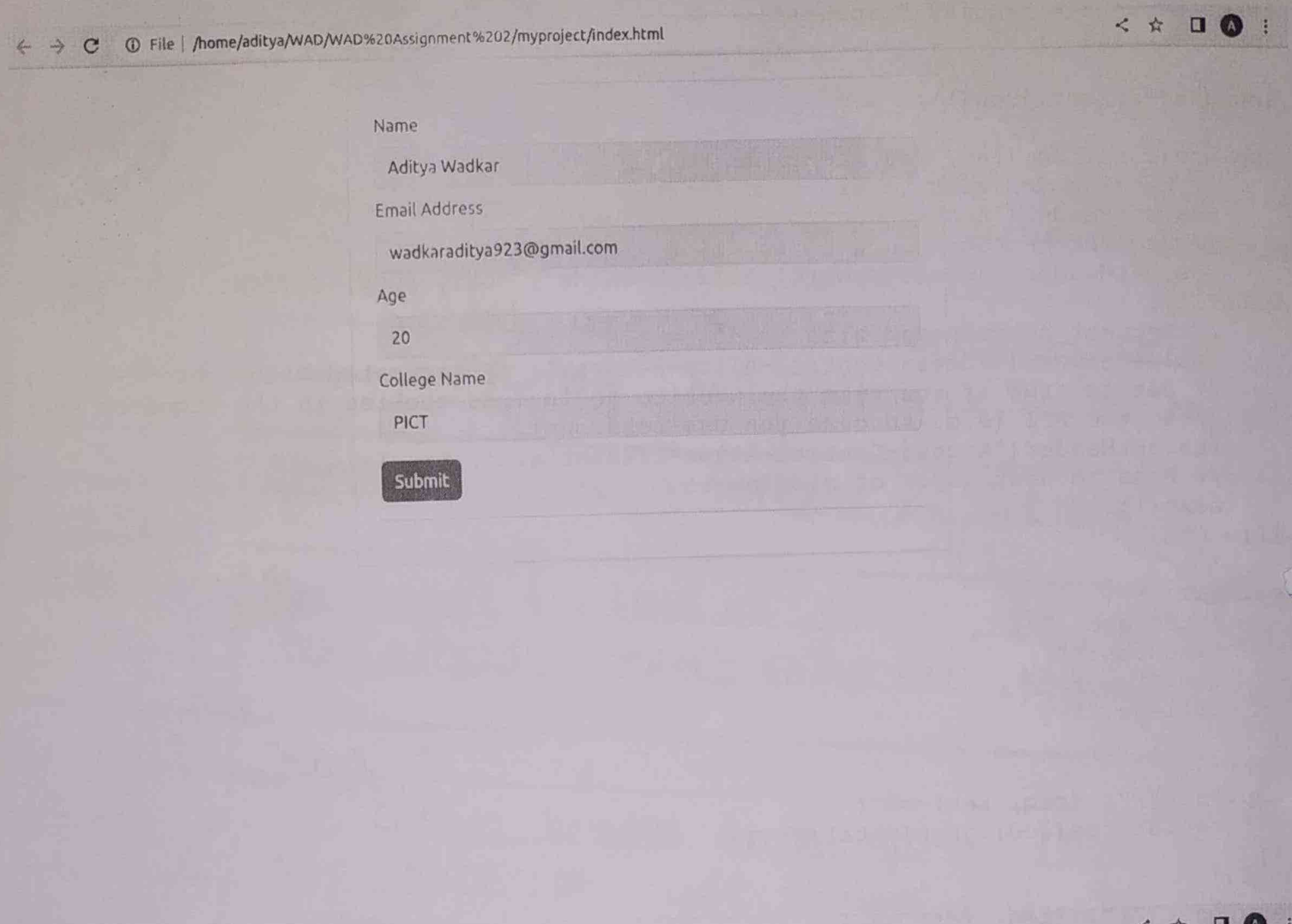
```
                var xhttp = new XMLHttpRequest();
                xhttp.onreadystatechange = function() {
                        if (this.readyState == 4 && this.status == 200) {
                                let data = JSON.parse(xhttp.responseText)

                                let string = `name: ${data.firstName} <br>email: ${data.Email} <br>age:
${data.age}<br>college: ${data.college}`;

                                w = window.open();
                                w.document.body.innerHTML = "<h1>User Details</h1><h3>";
                                w.document.body.innerHTML += string;
                                w.document.body.innerHTML += "</h3>";

                        }
                };
                xhttp.open("GET", "http://localhost:3000/", true);
                xhttp.send();
        }
        document.body.onload = () => {
                document.forms.addNumberForm.addEventListener("submit", (event) => {
                        event.preventDefault();
                        const firstName = event.target.firstName.value;
                        const Email = event.target.Email.value;
                        const age = event.target.age.value;
                        const college = event.target.college.value;

                        $.ajax({
                                type: 'post',
                                url: 'http://localhost:3000/',
                                data: JSON.stringify({
                                        firstName, Email, age, college
                                }),
                                contentType: "application/json; charset=utf-8",
                                traditional: true,
                                success: function(data) {
                                        fetchData();

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

</html>
```

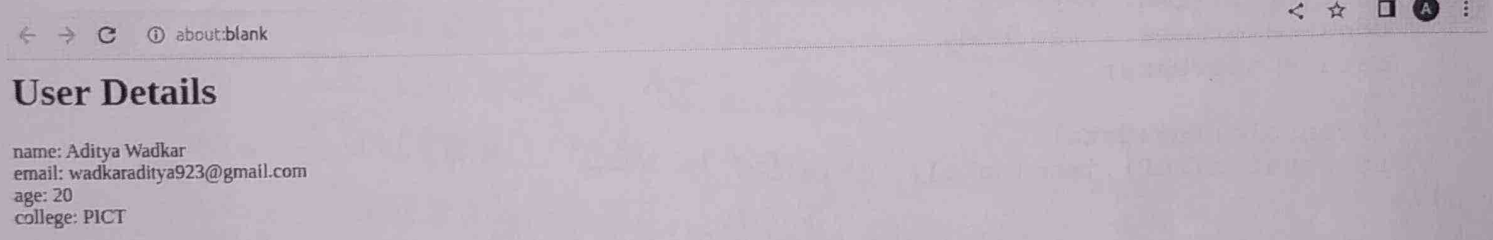
App.js

```
const express = require("express")
app = express();

app.use(express.json());

app.use(function (req, res, next) {
    // Website you wish to allow to connect
    res.setHeader('Access-Control-Allow-Origin', '*');
    // Request methods you wish to allow
    res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS, PUT, PATCH,
DELETE');
    // Request headers you wish to allow
    res.setHeader('Access-Control-Allow-Headers', 'X-Requested-With,content-type');
    // Set to true if you need the website to include cookies in the requests sent
    // to the API (e.g. in case you use sessions)
    res.setHeader('Access-Control-Allow-Credentials', true);
    // Pass to next layer of middleware
    next();
});

var data = {
    firstName: "",
    lastName:"",
    age:"",
    college:""
}

app.get("/", (req, res) => {
    res.status(200).json(data);
})

app.post("/", (req, res) => {
    const userData = req.body
    data = userData;

    //console.log(data)
    res.status(200).json(data);
});

app.listen(3000, () => {
    console.log("http://localhost:3000")
});
```



Output:


File | /home/aditya/WAD/WAD%20Assignment%202/myproject/index.html
Name
Aditya Wadkar
Email Address
wadkaraditya923@gmail.com
Age
20
College Name
PICT
Submit


about:blank
User Details
name: Aditya Wadkar
email: wadkaraditya923@gmail.com
age: 20
college: PICT

Name :- Aditya Anand Wadkar
Class :- TE I1 Batch - N7 I
Roll No: 33379

PICT, PUNE

Assignment 2a

Title :- create version control account on Github and using Git commands to create repository and push your code on Github

Objective :- create repositories on Github

S/w package & H/w apparatus :- Linux OS, VScode, PC with 128MB Ram, 40 GB HDD, keyboard mouse.

Theory :-

Git is version control system. It helps you keep track of code changes & it is used to collaborate on code.

It is used for :-

- Tracking code changes
- Tracking who made changes
- coding collaboration.

What does Git do?

- Manage project with Repositories
- Clone project to work on local copy
- Pull latest version of project to local copy
- Push local updates to main project.

Working with Git:

- Initialize ~~Git~~ Git on folder
- Git creates hidden folder to keep track.
- When file is deleted, changed, it considers modified.
- you select modified files you want to stage

- The staged files are commited
- Git allows you to see full history of every commit.
- you can revert back to any previous commit

What is Github?

- Git is not same as Github
- Github makes tools that use Git.

Steps to create version control account on Github and using Git command to create repository push your code to Github

- Sign up for Github Account
- Install Git.
- Create a new repository
- Initialize a Git repository
- Add files to repository
- commit changes
- Link your local repository to Github repo
- Push changes to Github.

Detailed steps:-

- Sign up for GitHub account.
  Go to Github website and signup for account by providing email, username & password. you can either free version or paid version of Github, depending on your needs.

Date: / /
Page No.:

- **Install Git**

Download and Install Git on your local machine if you haven't already. Git is a free and open-source version control system that allows you to manage:

- **Create new Repository:**

Login to your GitHub account and click on "new" button to create new repository. Give it a name and description, and choose whether it will be public or private.

- **Initialize Git Repository:**

In your local project directory, run command "git init" to initialize Git repository. This will create a hidden .git directory in your project.

- **Add files to repository:**

Use command "git add ." to add all files in your local project directory to git repository. Alternatively you can add individual files by using command "git add filename".

- **Commit changes:**

Use command: "git commit -m "commit message" to commit your changes to repo. The commit message should be brief description.

- **Link your local repository to your Github repo:**

Run command "git remote add origin https://github.com/your repo-name.git" to link your local Git repository to GitHub repository you

Date : / /
Page No.:

you created in step 3

- Push changes to GitHub:

Use command "git push -u origin master" to push changes in local repository to push changes In your local repository to Github repo. This will upload your code to github and make it available for other to see

Once you've completed these steps, you can continue to use Git and github to manage your code changes and collaborate with other.

Conclusion:

Git and GitHub provide a powerful and flexible platform for version control and collaborative s/w development By learning these basic steps, you can start to use Git and Github for your own projects and join a global community of developers who're these tools to build innovative s/w solution.

ASSIGNMENT 2a

Name: - Aditya Wadkar

Roll No: - 33379

Class: - TE-11

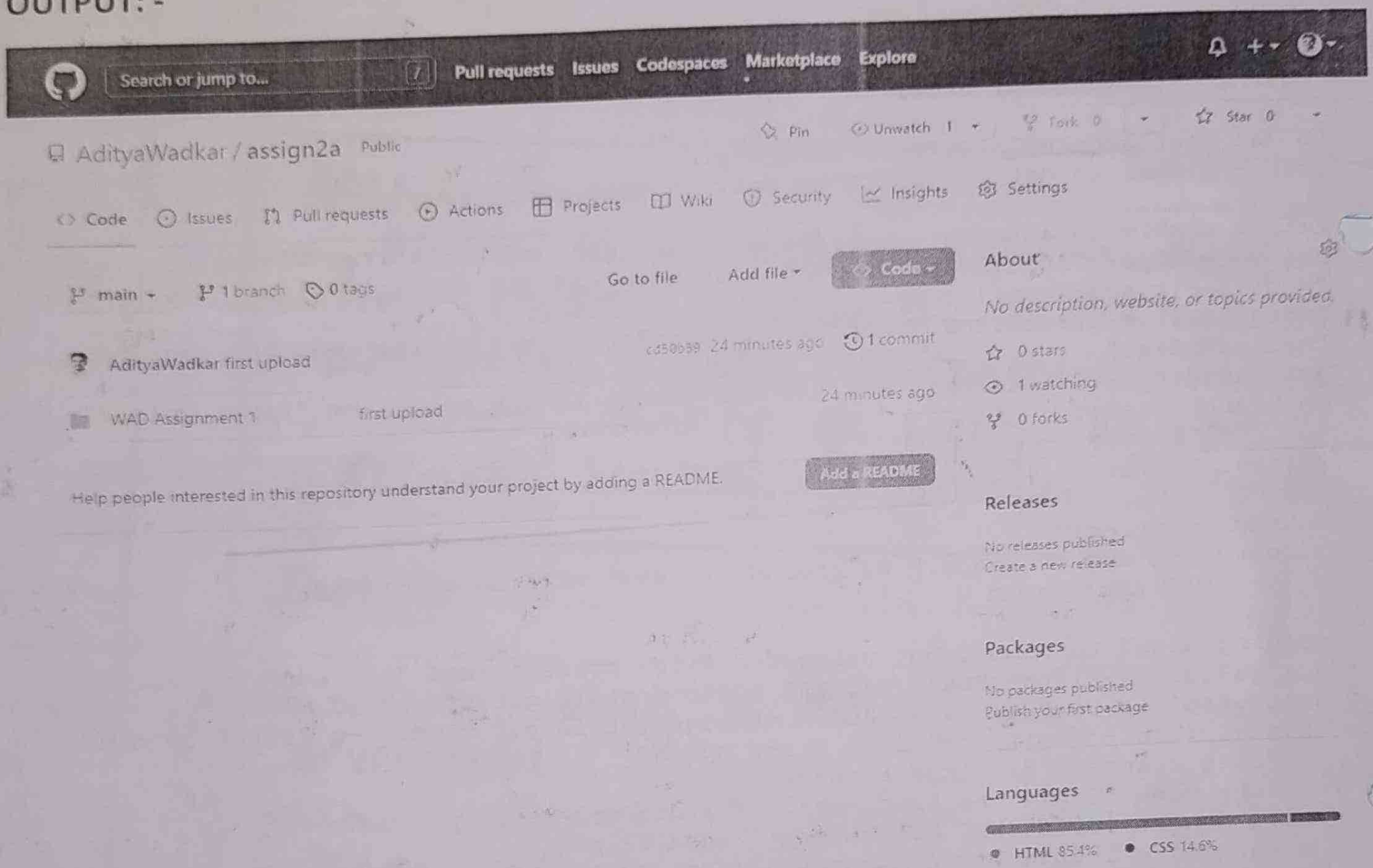
subject: - WADL

Terminal Commands

```
aditya@aditya-VirtualBox:~/WAD/WAD Assignment 1$ git init
Initialized empty Git repository in /home/aditya/WAD/WAD Assignment 1/.git/
aditya@aditya-VirtualBox:~/WAD/WAD Assignment 1$ git commit -m "first upload"

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'aditya@aditya-VirtualBox.(none)')
aditya@aditya-VirtualBox:~/WAD/WAD Assignment 1$ git config --global user.email
"wadkaraditya923@gmail.com"
aditya@aditya-VirtualBox:~/WAD/WAD Assignment 1$ git config --global user.name
"AdityaWadkar"
aditya@aditya-VirtualBox:~/WAD/WAD Assignment 1$ git commit -m "first upload"
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        WAD Assignment 1/

nothing added to commit but untracked files present (use "git add" to track)
aditya@aditya-VirtualBox:~/WAD/WAD Assignment 1$ git add .
aditya@aditya-VirtualBox:~/WAD/WAD Assignment 1$ git commit -m "first upload"
[master (root-commit) cd50b39] first upload
 7 files changed, 232 insertions(+)
 create mode 100644 WAD Assignment 1/cardio.png
 create mode 100644 WAD Assignment 1/cardio2.png
 create mode 100644 WAD Assignment 1/goal.png
 create mode 100644 WAD Assignment 1/index.html
 create mode 100644 WAD Assignment 1/progress.png
 create mode 100644 WAD Assignment 1/stats.png
 create mode 100644 WAD Assignment 1/style.css
aditya@aditya-VirtualBox:~/WAD/WAD Assignment 1$ git branch -M main
aditya@aditya-VirtualBox:~/WAD/WAD Assignment 1$ git remote add origin
https://github.com/AdityaWadkar/assign2a.git
aditya@aditya-VirtualBox:~/WAD/WAD Assignment 1$ git push -u origin main
Username for 'https://github.com': AdityaWadkar
Password for 'https://AdityaWadkar@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/en/get-started/getting-started-with-
git/about-remote-repositories#cloning-with-https-urls for information on currently
recommended modes of authentication.
fatal: Authentication failed for 'https://github.com/AdityaWadkar/assign2a.git/'
aditya@aditya-VirtualBox:~/WAD/WAD Assignment 1$ git push -u origin main
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
error: src refspec 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 does not match any
error: failed to push some refs to 'https://github.com/AdityaWadkar/assign2a.git'
aditya@aditya-VirtualBox:~/WAD/WAD Assignment 1$ git push -u origin main
Username for 'https://github.com': AdityaWadkar
```

```
Password for 'https://AdityaWadkar@github.com':
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 3 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (10/10), 152.00 KiB | 30.40 MiB/s, done.
Total 10 (delta 0), reused 0 (delta 0)
To https://github.com/AdityaWadkar/assign2a.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
aditya@aditya-VirtualBox:~/WAD/WAD Assignment 1$
```

OUTPUT: -



Name : Aditya wadkar

Class : TE I I Batch : N-I I

Roll NO : 33379

PICT, PUNE

Assignment 2b

Title :

Create Docker container Environment (NVIDEIA) Docker or other.

Docker :

Docker is a popular container platform that enables developers to package their application & their dependencies in to a container, making it highly portable & flexible solution for deploying software.

Features :

- Docker has ability to reduce size of the development.
- with containers, it becomes easier for teams across different unit working on DA, etc.
- you can deploy container anywhere where you wish.
- Docker is easily scalable.

components

- Docker for mac:
  For mac os docker
- Docker for linux
- Docker for linux
- Docker Engine
- Docker Hub
- Docker Compose.

Docker Installation

Step 1 : Ensure linux version 3.8 or higher by $ ~~user~~ uname command.

Step 2 : update OS with latest package
$ apt-get update.

Step 3 :- Install necessary certification
$ sudo apt-get install apt-transport-https ca-certificates.

Step 4 : Add new GPG key
$ sudo apt-key adv
--keyserver hkp://ha.pool.sks-keyservers.net
--rev-key 58118E89F3A912897C070ADD
BE76221572C5260GD

Step 5 :- Add relevant site to docker list for apt-package manager.

$ echo "deb https://apt.dockerproject.org | sudo tee /etc/apt/sources.list.d/docker.list

---

Step 6 : Apt-get update

Step 7 : verify package in pointing to right repository

$ apt-cache policy docker-engine.

Step 8 : $ apt-get update.

Step 9 : Install linux-image-extra kernel package

$ sudo apt-get install linux-image-extra-$

$ (uname -r) linux-image-extra-virtual.

Step 10 : Install Docker

$ sudo apt-get install -y docker-engine.

---

* creating a Docker container environment.

1. Installing Docker
2. creating a Docker file : It is a text file containing instruction for building a Docker Image.
3. Building Docker Image : you can use "Docker build" command to build Docker Image.
4. Running Docker environment use "Docker run" to start new container from image file.

---

conclusion.

Hence we successfully have install Docker & created a container using it.

9/3/23

Assignment 2b

Name :- Aditya Wadkar
Class :- TE-11

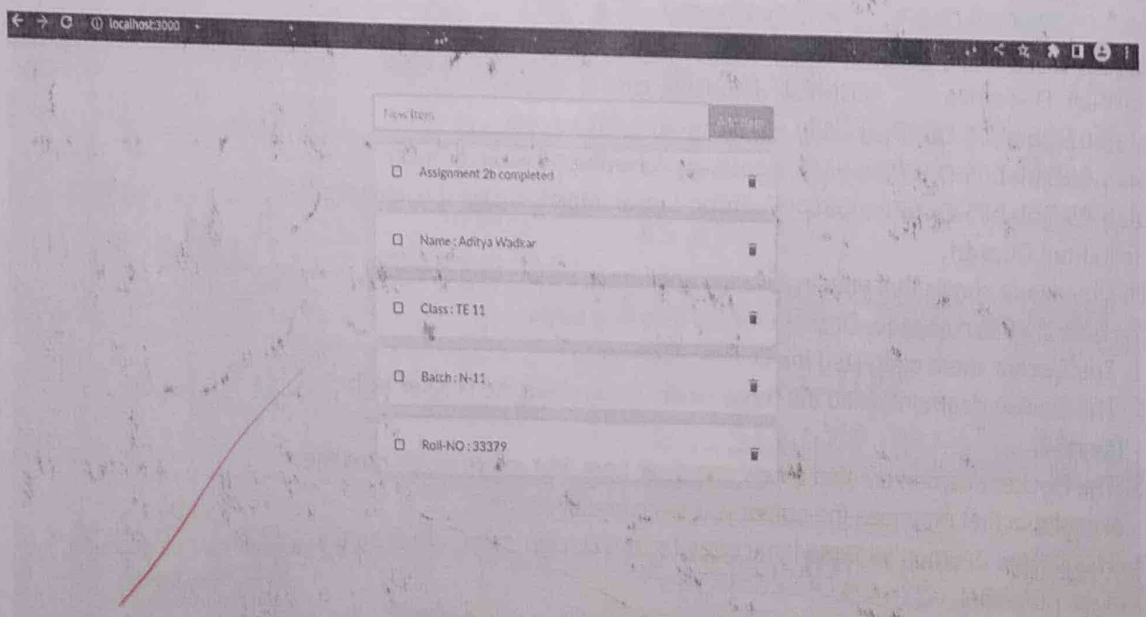
Roll No :- 33379
Subject :- WADL

Terminal Commands :

```
blab-05@blab05-OptiPlex-5060:~$ sudo apt-get update
sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
blab-05@blab05-OptiPlex-5060:~$ sudo mkdir -m 0755 -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
blab-05@blab05-OptiPlex-5060:~$ echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
blab-05@blab05-OptiPlex-5060:~$  sudo apt-get install docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin
blab-05@blab05-OptiPlex-5060:~$ sudo usermod -aG docker $USER
[sudo] password for blab-05:
blab-05@blab05-OptiPlex-5060:~$ git clone https://github.com/docker/getting-started.git
Cloning into 'getting-started'...
remote: Enumerating objects: 952, done.
remote: Total 952 (delta 0), reused 0 (delta 0), pack-reused 952
Receiving objects: 100% (952/952), 5.18 MiB | 1.27 MiB/s, done.
blab-05@blab05-OptiPlex-5060:~$ cd getting-started
blab-05@blab05-OptiPlex-5060:~/getting-started$ ls
app       docker-compose.yml  docs     mkdocs.yml  requirements.txt
build.sh  Dockerfile          LICENSE  README.md
blab-05@blab05-OptiPlex-5060:~/getting-started$ cd app
blab-05@blab05-OptiPlex-5060:~/getting-started/app$ newgrp docker
blab-05@blab05-OptiPlex-5060:~/getting-started/app$ docker run hello-world
Hello from Docker!
This message shows that your installation appears to be working correctly.
To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.
To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash
```

```
Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/
For more examples and ideas, visit:
 https://docs.docker.com/get-started/
blab-05@blab05-OptiPlex-5060:~/getting-started/app$ docker build -t getting-started .
[+] Building 73.0s (11/11) FINISHED
 => [internal] load .dockerignore                                    0.2s
 => => transferring context: 2B                                      0.0s
 => [internal] load build definition from Dockerfile                 0.3s
 => => transferring dockerfile: 185B                                 0.0s
 => resolve image config for docker.io/docker/dockerfile:1           4.4s
 => docker-image://docker.io/docker/dockerfile:1@sha256:39b85bbfa7536a5fe  8.4s
 => => resolve docker.io/docker/dockerfile:1@sha256:39b85bbfa7536a5feceb7  0.5s
 => => sha256:966d40f9ba8366e74c2fa353fc0bc7bbc167d2a0f3ad242 482B / 482B  0.0s
 => [internal] load build context                                    0.7s
 => => transferring context: 4.59MB                                  0.0s
 => [2/4] WORKDIR /app                                               2.0s
 => [3/4] COPY . .                                                   0.8s
 => [4/4] RUN yarn install --production                             14.3s
 => exporting to image                                               3.4s
 => => exporting layers                                              3.3s
 => => writing image sha256:3838b863c5735e31cec4a2821d5fd861d68e5443c6860  0.0s
 => => naming to docker.io/library/getting-started                   0.0s
blab-05@blab05-OptiPlex-5060:~/getting-started/app$ docker run -dp 3000:3000 getting-started
3ee52a852a43eba5c50986aabe46a27671cf22646df949b0a7b2857396f56919
blab-05@blab05-OptiPlex-5060:~/getting-started/app$
```

Output :-



PICT, PUNE

Name: Aditya Wadkar
Class: TE I I Roll NO: 33379

Assignment 2C

Title: Angular Application

Problem Statement:
Create an Angular application which will do following action: Register user, Login user, show lines data on Profile component

Theory:

① Angular require a current, active version of Node.js & NPM
Install Node JS.
It will automatically intall NPM node package manager.

② Install Angular CLI
npm install -g @angular/cli@latest

③ To create new project through CLI go to folder of new project & write
-ng new projectname.

④ open src/app

⑤ write HTML code

⑥ make changes in app.js.

step ①: create an angular application
ng new my-app

step ②: create new service to handle user related functionalities.

step ③: define user model to represent user data

step ④: create a registration component to allow register for account.

step ⑤: create login component to allow user to login their account

step ⑥: create a profile component to display user data.

step (7) : creating a routing model to define routes.

step (8) : create navigation bar to allow user to navigate bet~ different components

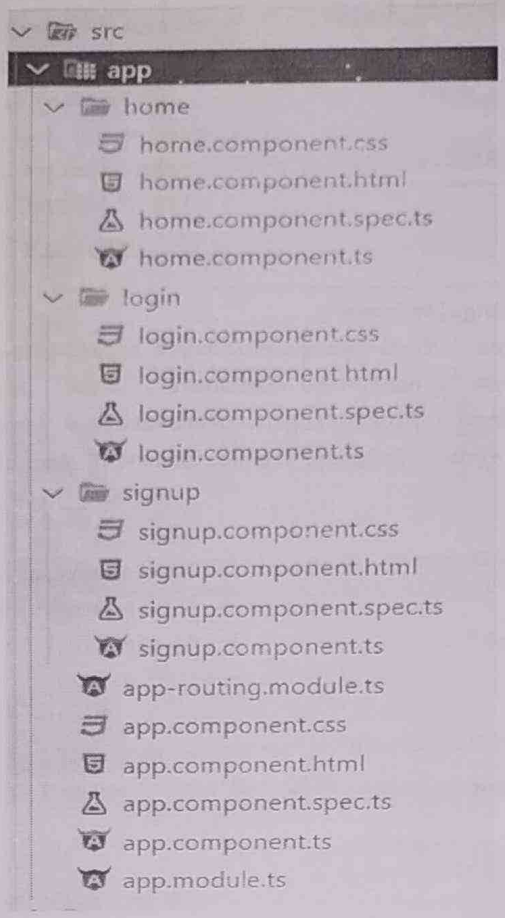
step (9) : update the app component to include Routing & user service

---

conclusion :-

Hence we have successfully have created an Angular Application.

Directory Structure



index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>MyAssignment</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
aFq/bzH65dt+w6FI2ooMVUpc+21e0SRygnTpmBvdBgSdnuTN7QbdgL+OapgHtvPp"
crossorigin="anonymous">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
```

```
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'my-assignment';
}
```

app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { LoginComponent } from './login/login.component';
import { SignupComponent } from './signup/signup.component';

const routes: Routes = [
  {
    path: "",
    component: HomeComponent
  },
  {
    path: "login",
    component: LoginComponent
  },
  {
    path: "signup",
    component: SignupComponent
  },
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

app.component.html

```
<router-outlet></router-outlet>
```

signup.component.ts

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
@Component({
    selector: 'app-signup',
    templateUrl: './signup.component.html',
    styleUrls: ['./signup.component.css']
})
export class SignupComponent {
    usernameError: string = "";
    passwordError: string = "";
```

```
    constructor(private router: Router) { }
    async getData(username: any, password: any) {
        this.usernameError = username ? "" : "Username is required";
        this.passwordError = password ? "" : "Password is required";
        if (username && password) {
            var myHeaders = new Headers();
            myHeaders.append("Content-Type", "application/json");
            var raw = JSON.stringify({
                "username": username,
                "password": password
            });
            var requestOptions: Object = {
                method: 'POST',
                headers: myHeaders,
                body: raw,
                redirect: 'follow'
            };
            fetch("http://localhost:2324/signup", requestOptions)
                .then(response => {
                    if (response.ok) {
                        return response.json();
                    } else {
                        this.usernameError = "Username Already Exists";
                        throw Error("Not OK")
                    }
                })
                .then(result => {
                    if (result.token) {
                        document.cookie = "jwt=" + result.token
                        localStorage.setItem("user", JSON.stringify(result.user))
                        this.router.navigate(['']);
                    }
                })
                .catch(error => console.log('error', error));
        }
    }
}
```

signup.component.html

```
<div class="container py-5">
    <div class="col-12 col-sm-10 col-md-6 col-lg-4 m-auto border shadow rounded p-4">
        <h3 class="mb-4 pb-2 border-bottom">User Signup</h3>
        <div class="mb-3">
            <label for="username" class="form-label">username</label>
            <input type="username" class="form-control" name="username" #username aria-
describedby="usernameHelpId" placeholder="abc">
            <small id="usernameHelpId" class="form-text text-
danger">{{usernameError}}</small>
        </div>
        <div class="mb-3">
```

```
            <label for="" class="form-label">Password</label>
            <input type="password" class="form-control" name="password" #password aria-
describedby="passwordHelpId" placeholder="Your Password">
            <small id="passwordHelpId" class="form-text text-
danger">{{passwordError}}</small>
        </div>
        <button type="button" class="btn btn-primary" #loginBtn
(click)="getData(username.value, password.value)">Login</button>
        <br>
        <hr />
        <a href="/login">Already Have An Account?</a>
    </div>
</div>
```

login.component.ts

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
@Component({
    selector: 'app-login',
    templateUrl: './login.component.html',
    styleUrls: ['./login.component.css']
})
export class LoginComponent {
    usernameError: string = "";
    passwordError: string = "";
    constructor(private router: Router) {
    async getData(username: any, password: any) {
        this.usernameError = username ? "" : "Username is required";
        this.passwordError = password ? "" : "Password is required";
        if (username && password) {
            var myHeaders = new Headers();
            myHeaders.append("Content-Type", "application/json");
            var raw = JSON.stringify({
                "username": username,
                "password": password
            });
            var requestOptions: Object = {
                method: 'POST',
                headers: myHeaders,
                body: raw,
                redirect: 'follow'
            };
            fetch("http://localhost:2324/login", requestOptions)
                .then(response => response.json())
                .then(result => {
                    if (result.token) {
                        document.cookie = "jwt=" + result.token
                        localStorage.setItem("user", JSON.stringify(result.user))
                        this.router.navigate(['']);
                    } else {
```

```
                        this.usernameError = result.message == "Username Not Found" ?
result.message : "";
                        this.passwordError = result.message == "Incorrect Password" ?
result.message : "";
                    }
                })
                .catch(error => console.log('error', error));
        }
    }
}
```

login.component.html

```
<div class="container py-5">
    <div class="col-12 col-sm-10 col-md-6 col-lg-4 m-auto border shadow rounded p-4">
        <h3 class="mb-4 pb-2 border-bottom">User Login</h3>
        <div class="mb-3">
            <label for="username" class="form-label">username</label>
            <input type="username" class="form-control" name="username" #username aria-
describedby="usernameHelpId" placeholder="abc">
            <small id="usernameHelpId" class="form-text text-
danger">{{usernameError}}</small>
        </div>
        <div class="mb-3">
            <label for="" class="form-label">Password</label>
            <input type="password" class="form-control" name="password" #password aria-
describedby="passwordHelpId" placeholder="Your Password">
            <small id="passwordHelpId" class="form-text text-
danger">{{passwordError}}</small>
        </div>
        <button type="button" class="btn btn-primary" #loginBtn
(click)="getData(username.value, password.value)">Login</button>
        <br>
        <hr />
        <a href="/signup">Create An Account?</a>
    </div>
</div>
```

home.component.ts

```
import { Component } from '@angular/core';
import { CookieService } from 'ngx-cookie-service';
import { Router } from '@angular/router';

@Component({
    selector: 'app-home',
    templateUrl: './home.component.html',
    styleUrls: ['./home.component.css']
})
export class HomeComponent {
    message: string
    constructor(private cookieService: CookieService, private router: Router) {
        let user: string = localStorage.getItem("user") ?? "{}";
```

```
        const userDetails = JSON.parse(user)
        this.message = userDetails.username ? "Hello " + userDetails.username : "LOGIN";
        this.validate()
    }
    validate() {
        // verify
        const jwt = this.cookieService.get('jwt');
        console.log(jwt)
        const myHeaders = new Headers({
            'Content-Type': 'application/json',
            'Cookie': `jwt=${jwt}`
        });
        var requestOptions: Object = {
            method: 'GET',
            headers: myHeaders,
            redirect: 'follow',
            credentials: 'include'
        };
        fetch("http://localhost:2324", requestOptions)
            .then(response => {
                if (!response.ok) {
                    throw new Error(String(response.status));
                }
                return response.json();
            })
            .then(result => {
                console.log(result)
                if (result.message == "Token is not valid") {
                    // redirect to login
                    localStorage.removeItem("user")
                    this.router.navigate(['/login']);
                }
                else if (result.message == "Token is not provided") {
                    // redirect to login
                    localStorage.removeItem("user")
                    this.router.navigate(['/login']);
                }
                if (this.message == "LOGIN") {
                    this.router.navigate(['/login']);
                }
            })
            .catch(error => console.log('error', error));
    }
    logout() {
        localStorage.removeItem("user")
        this.router.navigate(['/login']);
    }
}
```

home.component.html

```
<p class="text-center h1 m-5">{{message}}</p>
<div class="text-center">
```

```
    <button class="btn btn-danger my-5" (click)="logout()">LogOUT</button>
</div>
```

BACKEND: index.js

```
import express from "express"
import User from "./User.js"
import cookieParser from "cookie-parser";
import jwt from "jsonwebtoken";
import connection from "./mongodb.js";
import cors from 'cors';
import dotenv from "dotenv"
dotenv.config();
const app = express();
//middleware
app.use(express.json());
app.use(cookieParser());
app.use(function (req, res, next) {
    res.setHeader('Access-Control-Allow-Origin', '*');
    res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS, PUT, PATCH,
DELETE');
    res.setHeader('Access-Control-Allow-Headers', 'X-Requested-With,content-type');
    res.setHeader('Access-Control-Allow-Credentials', true);
    next();
});
app.use(cors({
    origin: 'http://localhost:4200',
    credentials: true
}));
const authMiddleware = (req, res, next) => {
    const jwToken = req.cookies.jwt
    if (jwToken) {
        jwt.verify(jwToken, "This is My Secret", (error, decode) => {
            if (error) {
                res.json({ message: "Token is not valid" })
            } else {
                req._id = decode._id;
                next()
            }
        })
    } else {
        res.json({ message: "Token is not provided" })
    }
}
// functions
async function createToken(_id, res) {
    const token = await jwt.sign({ _id }, "This is My Secret", {
        expiresIn: "2d"
    })
    res.cookie("jwt", token, {
        maxAge: 1000 * 60 * 60 * 24 * 2,
        httpOnly: true
    });
```

```
    return token;
}
app.post("/login", async (req, res) => {
    try {
        const user = await User.login(req.body);
        if (user == 1) {
            res.clearCookie("jwt");
            return res.send({ message: "Username Not Found" })

        } else if (user == 2) {
            res.clearCookie("jwt");
            return res.send({ message: "Incorrect Password" })
        }

        const token = await createToken(user._id, res)
        return res.send({ user, token });
    } catch (error) {
        res.status(404).send({ error })
    }
})
app.post("/signup", async (req, res) => {
    const { username, password } = req.body;
    if (username && password) {
        try {
            const user = await User.create({ username, password });
            const token = await createToken(user._id, res)
            return res.send({ user, token });
        } catch (error) {
            return res.status(404).send({ error })
        }
    } else {
        return res.status(404).json({ message: "`username` and `password` are required"
})
    }
})

app.get("/", authMiddleware, async (req, res) => {
    try {
        const user = await User.findOne({ _id: req._id })
        res.status(200).json({ username: user.username })
    } catch (error) {
        res.status(494).json({ error })
    }
})
connection.then(() => {
    app.listen(2324, () => {
        console.log("server started: http://localhost:2324")
    })
}).catch(err => {
    console.log(err)
})
```

BACKEND: User.js

```
import mongoose from "mongoose"
import bcrypt from "bcrypt"
const User = new mongoose.Schema({
    username: {
        type: String,
        unique: true,
        trim: true,
        required: true
    },
    password: {
        type: String,
        trim: true,
        required: true
    }
});
User.pre("save", async function (next) {
    this.password = await bcrypt.hash(this.password, 10);
    next();
});
User.statics.login = async function ({ username, password }) {
    try {
        const user = await this.findOne({ username })
        if (user) {
            return await bcrypt.compare(password, user.password) ? user : 2
        } else {
            return 1;
        }
    } catch (error) {
        console.log(error)
    }
}
export default mongoose.model("user", User);
```

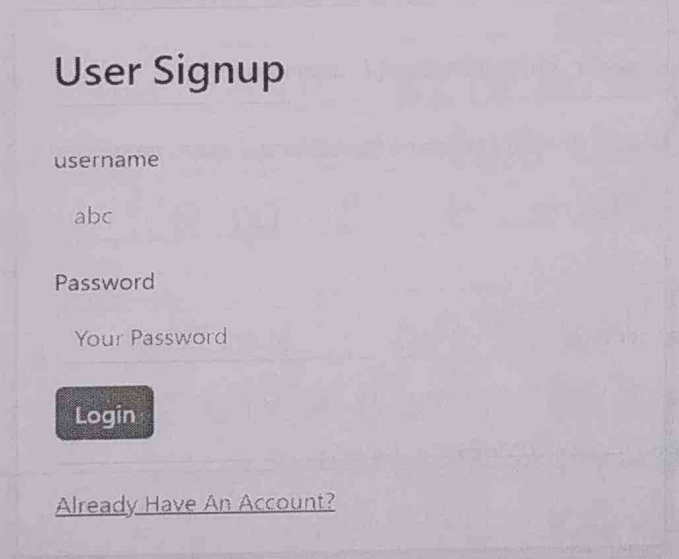
BACKEND:mongodb.js

```
import mongoose from "mongoose";
import dotenv from "dotenv"
dotenv.config();

const connection = mongoose.connect(process.env.DB_URI)
export default connection;
```

OUTPUT:

Login Page:

Signup Page:



Home Page (After Login/Signup):

Name: Aditya wadkar
Class : TE IT
Roll No: 33379

PICT, PUNE

Assignment 3a

Title: Static web server

Problem statement:
Create Node JS application which serves a static website

Objective: use necessary modules like Http, URL, Path etc for creation of server & file operation on server side.

Theory:
Here we will build a static file web server which will list files in directory and on clicking file name it display file content.

Step 1: Importing necessary modules and defining multipurpose Internet mail extension which helps browser to understand type of file that is being sent.
Step 2: creating a server at port specified.
Step 3: we will respond URL "/" to all list files in directory
Step 4: preprocessing required files pathname to avoid directory traversal.
Step 5: finally check whether the file exist ~~found~~ with 404 status code.

Else if not exist then send file NOT found.
with 404 status code.

Step 5 :

Finally, check whether the file exits. If exist then send then file, with the proper header content-type' having value as per file extension mapped with the MIME type above.

Conclusion :-

Thus, we have used http, URL, path node js modules to create static file web server.

Assignemnt 3A

Name: Aditya wadkar
Roll no.: 33379
Batch: N-11

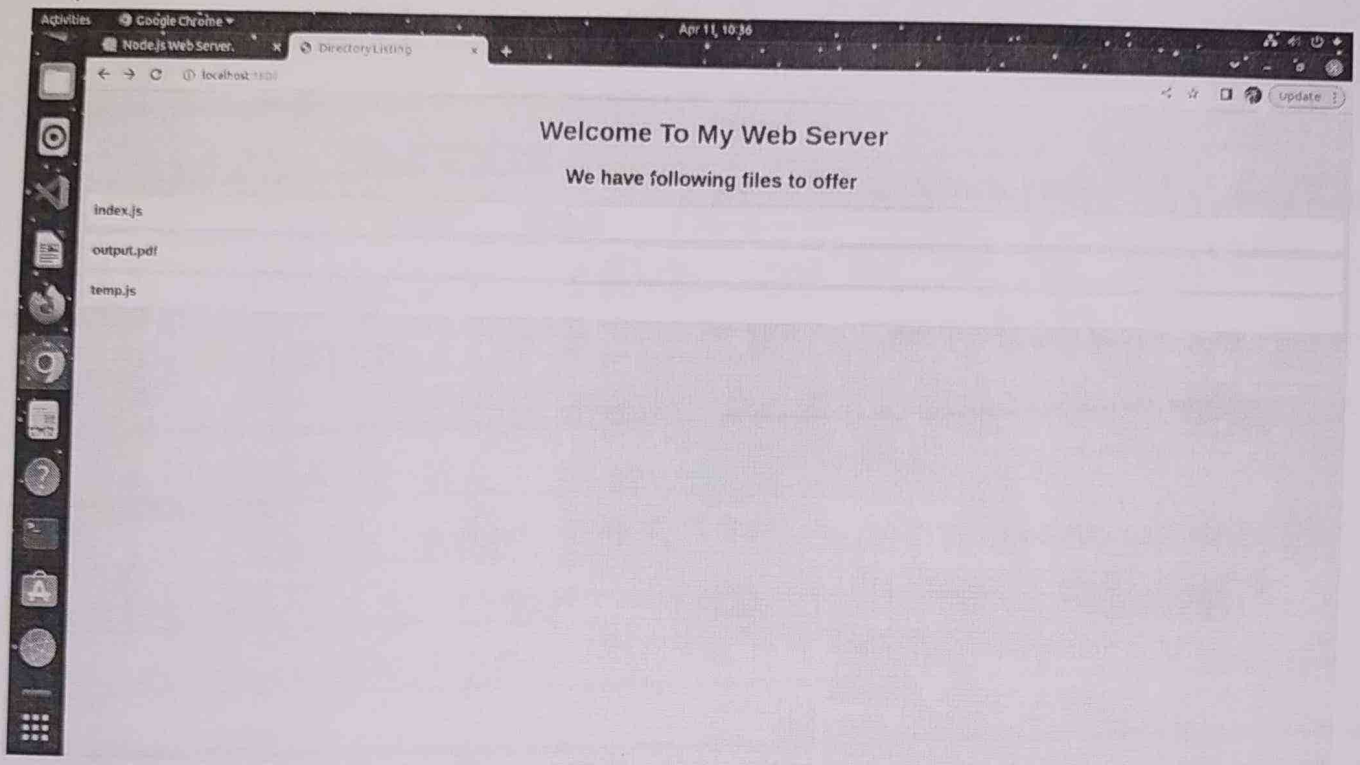
Index.html

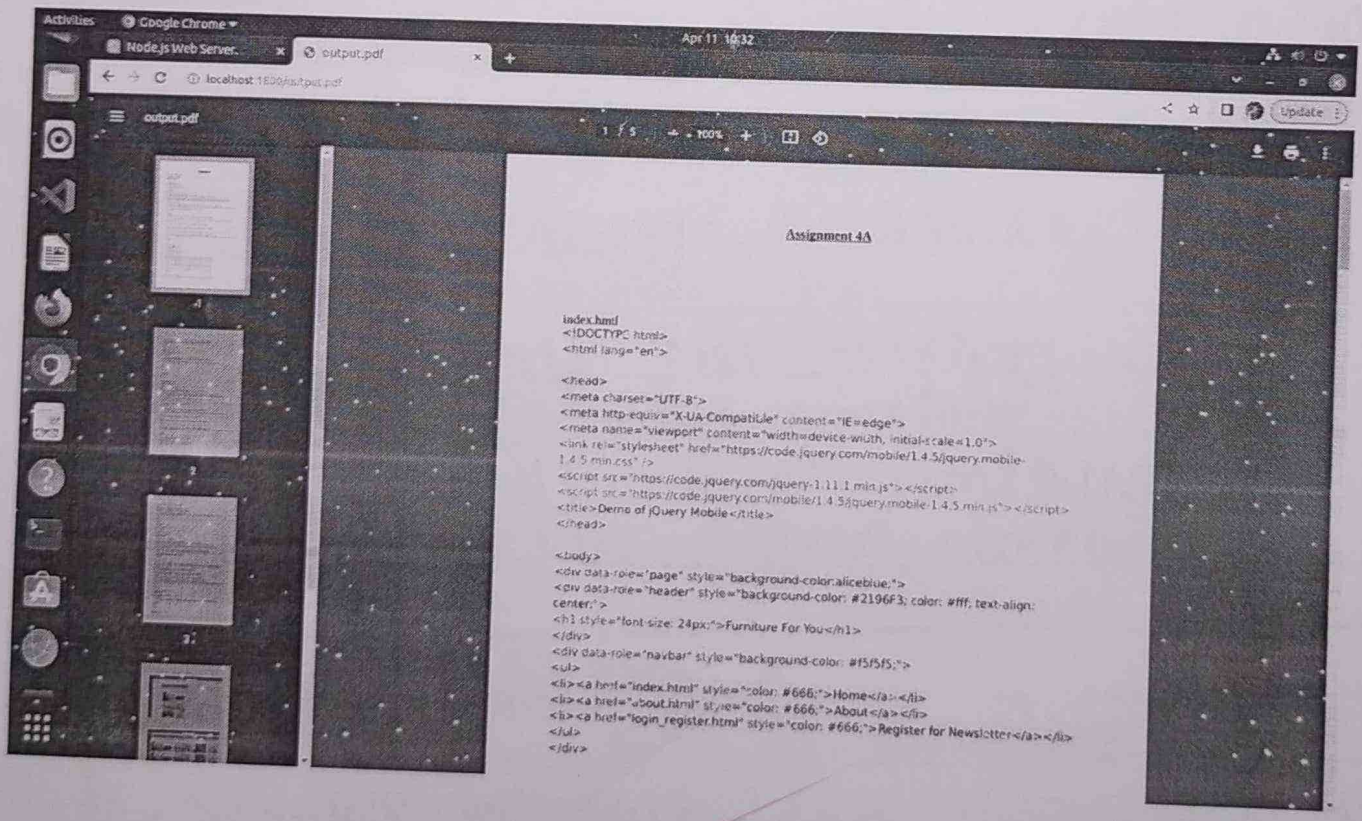
```
const http = require('http');
const url = require('url');
const fs = require('fs');
const path = require('path');

const PORT = 1800;

const mimeType = {
'.ico': 'image/x-icon',
'.html': 'text/html',
'.js': 'text/javascript',
'.json': 'application/json',
'.css': 'text/css',
'.png': 'image/png',
'.jpg': 'image/jpeg',
'.wav': 'audio/wav',
'.mp3': 'audio/mpeg',
'.svg': 'image/svg+xml',
'.pdf': 'application/pdf',
'.doc': 'application/msword',
'.eot': 'application/vnd.ms-fontobject',
'.ttf': 'application/font-sfnt'
};

http.createServer((req, res) => {
const parsedUrl = url.parse(req.url);
if (parsedUrl.pathname === '/') {
var filesList = fs.readdirSync("./");
var filesLink = "<ul>";
filesList.forEach(element => {
if (fs.statSync("./" + element).isFile()) {
filesLink += `<li><a href='./${element}'>${element}</a></li>`;
}
});
filesLink += "</ul>";
var html = `
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Directory Listing</title>
<style>
body {
```

```
font-family: Arial, Helvetica, sans-serif;
background-color: #f4f4f4;
}
h1 {
text-align: center;
color: #333;
}
ul {
list-style: none;
padding: 0;
margin: 0;
}
li {
margin-bottom: 10px;
background-color: #fff;
padding: 10px;
border-radius: 5px;
}
a {
color: #333;
text-decoration: none;
font-weight: bold;
}
a:hover {
text-decoration: underline;
}
</style>
</head>
<body>
<h1>Welcome To My Web Server <br></h1><h2 style="text-align: center"> We have following files to
offer</h2>
${filesLink}
</body>
</html>
`;
res.setHeader('Content-type', 'text/html');
res.end(html);
} else {
const sanitizePath = path.normalize(parsedUrl.pathname).replace(/^(\.\.[\/\\])+/, '');
let pathname = path.join(__dirname, sanitizePath);
if (!fs.existsSync(pathname)) {
res.statusCode = 404;
res.end(`File ${pathname} not found!`);
} else {
fs.readFile(pathname, function (err, data) {
if (err) {
res.statusCode = 500;
res.end(`Error in getting the file.`);
} else {
const ext = path.parse(pathname).ext;
res.setHeader('Content-type', mimeType[ext] || 'text/plain');
```

```
 res.end(data);
 }
 });
 }
 }
}).listen(PORT);
```

output:





Name: Aditya Wadkar
Class: TE I1
Roll no: 33379

PICT, PUNE

Assignment 3(b)

Title: API for CRUD operations.

Problem statement:
Create four API using Node JS, Express JS, MongoDB CRUD operation for Register user, Login user & show user data.

Objective:
Use node JS, express, mongoDB & mongoose libraries to create CRUD operation for user administration.

Theory:
CRUD is an acronym for, Creat, Read, update and delete.
It is a set of operation we get servers. to execute (PUST, GET, PUT & DELETE request).

CREATE (POST) - make something
READ (GET) - Get something
UPDATE (PUT) - Change something
DELETE (DELETE) - Remove something

Creating Application
- Create a new folder for application
- Initialize application with a package.json file.
- go to root folder of your application and type npm init to initialize your app with a package.json.
- Install Express, mongoose and body-parser modules

- create a new file new file named server.js in root.

Configuring and connecting to Database
- create a file database.config inside config

Create mongoose model
- create a folder called model inside app folder. create user.js & paste below code.

Create controller

Write following CRUD functions.
- create findall - findone - update - destory
- find a single user with an id
- update a user by id in request
- delete a user with specified id with request

* Creating a new user:
- create & save new user
- Retrive all users from database
- Retrieve all and update user by id
- Delete a user with specified in request
- Define Routes

Conclusion:

Thus, we have used node JS, express, mongo DB & mongoose initialized to create CRUD operation.

Assignment 3B

Name: Aditya wadkar
RollNo: 33379
Batch: N-11

Index: server.js

```
const express = require("express");
const bodyParser = require("body-parser");


app.use(bodyParser.urlencoded({ extended: true }));

app.use(bodyParser.json());

const UserRoute = require("./routes/User");
app.use("/user", UserRoute);

const dbConfig = require("./config/database.config.js");
const mongoose = require("mongoose");

mongoose.Promise = global.Promise;
mongoose
  .connect(dbConfig.url, {
    useNewUrlParser: true,
  })
  .then(() => {
    console.log("Databse Connected Successfully!!");
  })
  .catch((err) => {
    console.log("Could not connect to the database", err);
    process.exit();
  });
app.get("/", (req, res) => {
  res.json({ message: "Hello Crud Node Express" });
});
app.listen(3000, () => {
  console.log("Server is listening on port 3000");
});
```

Routes: User.js

```
const express = require("express");
const UserController = require("../controllers/User");
const router = express.Router();

router.get("/", UserController.findAll);
router.get("/:id", UserController.findOne);
router.post("/", UserController.create);
router.patch("/:id", UserController.update);
```

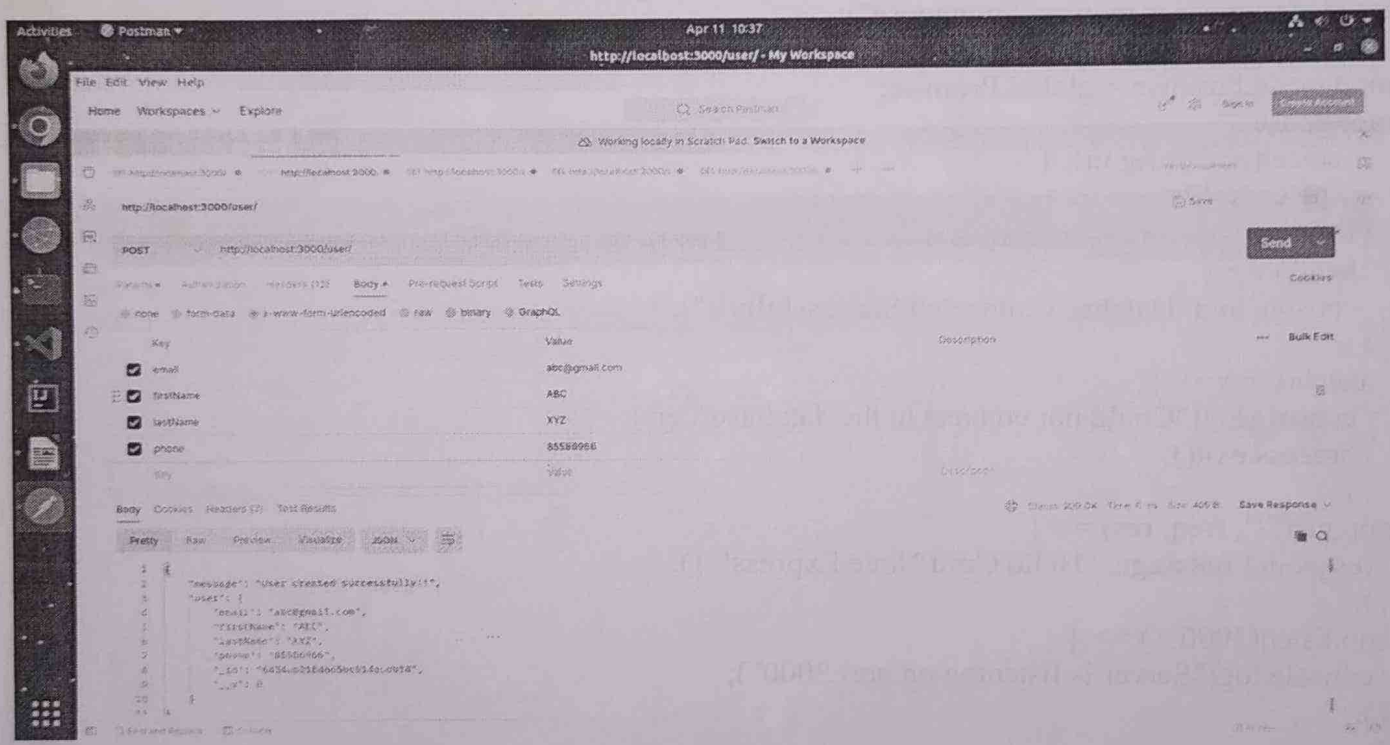
```
router.delete("/:id", UserController.destroy);

module.exports = router;
```

Model: User.js

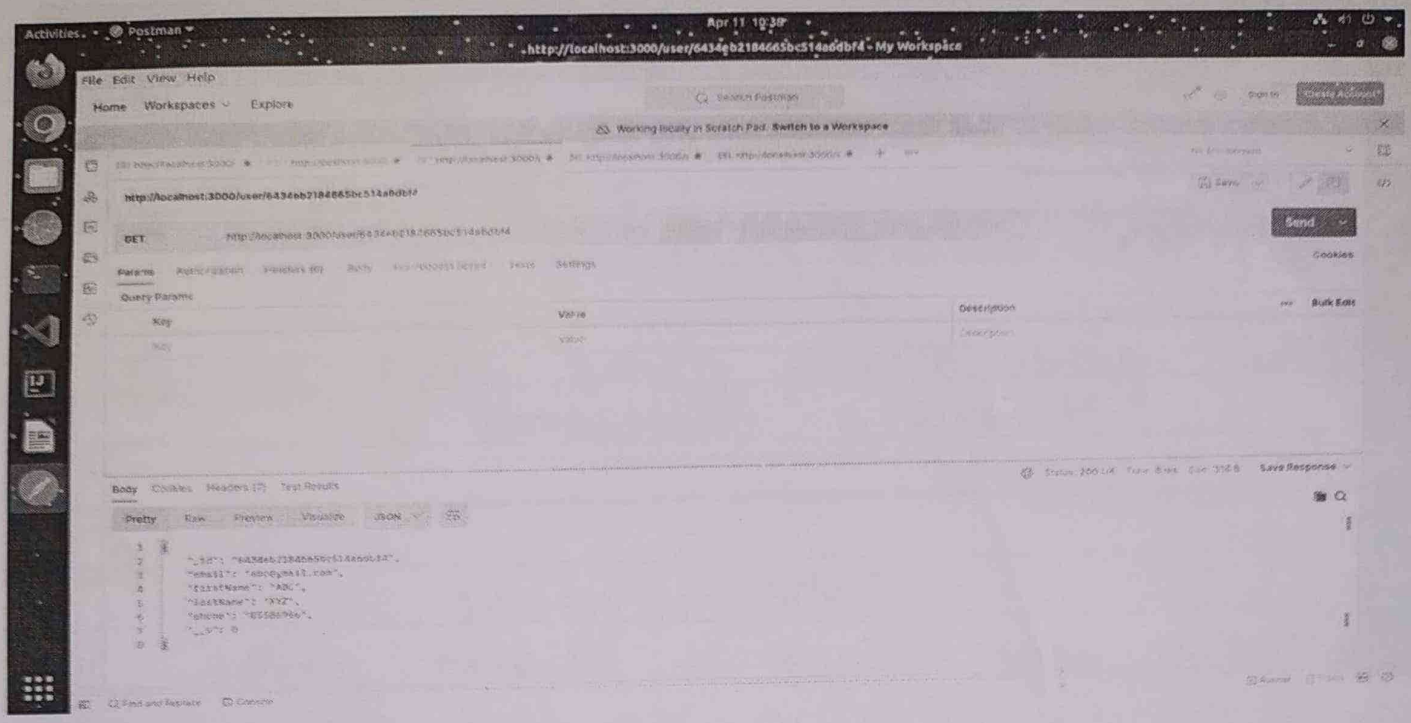
```
var mongoose = require("mongoose");
var schema = new mongoose.Schema({
  email: { type: String, required: true, unique: true },
  firstName: { type: String, default: "" },
  lastName: { type: String, default: "" },
  phone: String,
});
var user = new mongoose.model("User", schema);
module.exports = user;
```

Output:

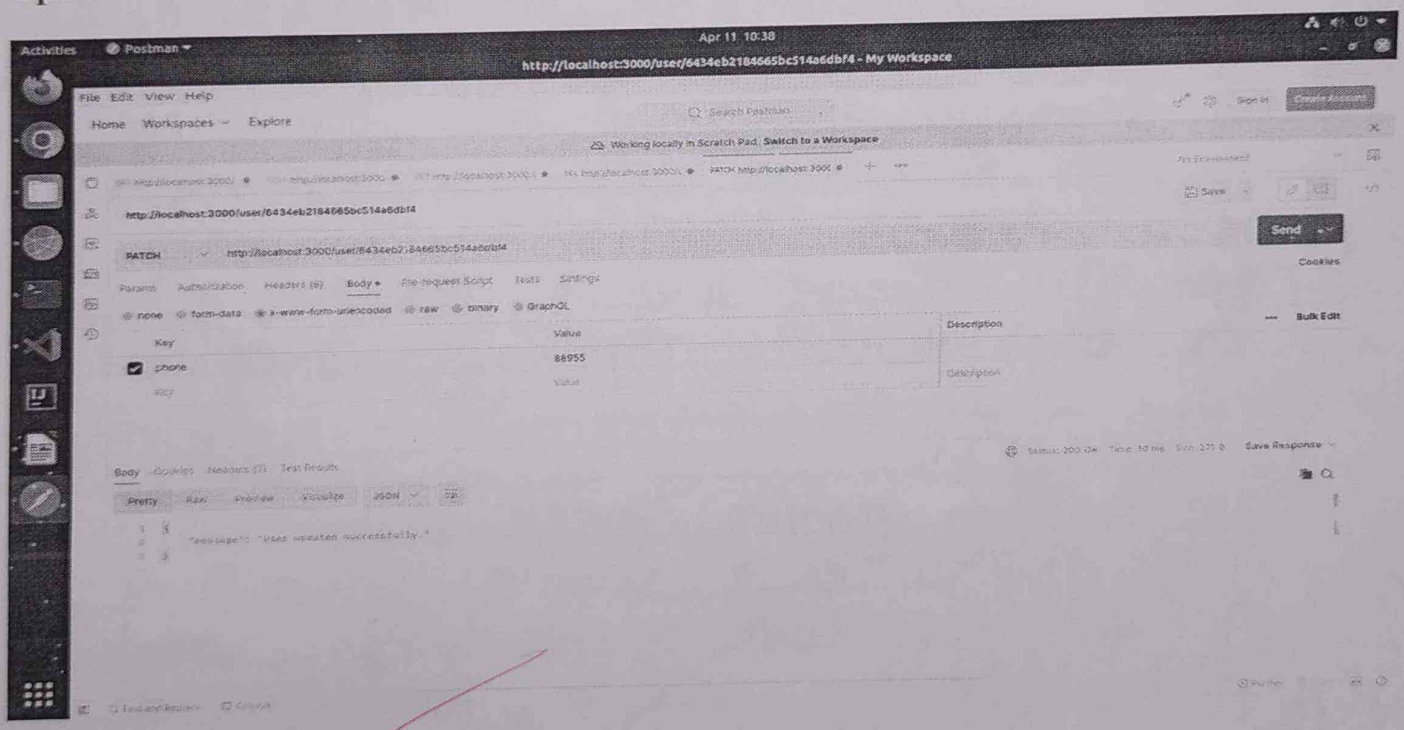
Create User:



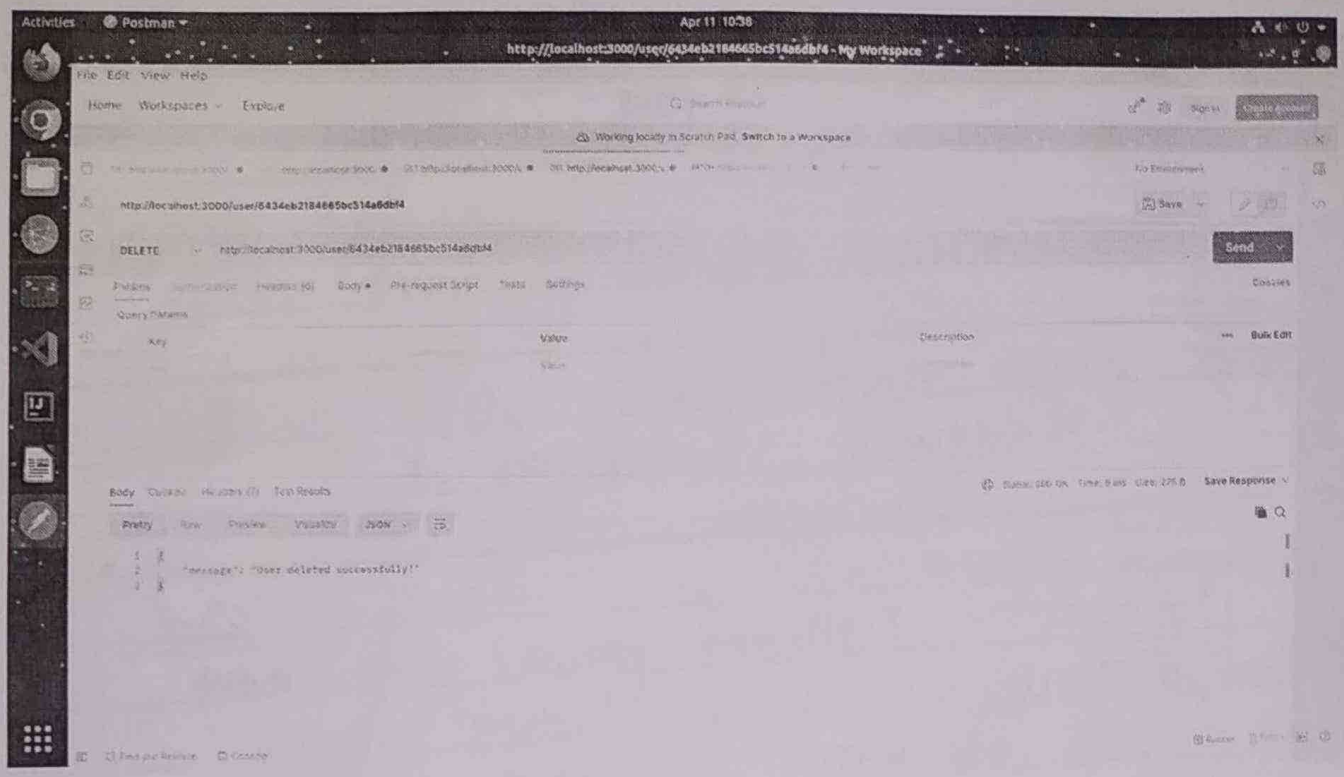
Display Single User:



Update User:



Delete User:


Activities
Postman
Apr 11 10:38
http://localhost:3000/user/6434eb2184665bc514a6dbf4 - My Workspace
File Edit View Help
Home Workspaces Explore
Working locally in Scratch Pad. Switch to a Workspace
http://localhost:3000/user/6434eb2184665bc514a6dbf4
Save
DELETE
http://localhost:3000/user/6434eb2184665bc514a6dbf4
Send
Params
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Pretty
Raw
Preview
Visualize
JSON
Save Response
{
"message": "User deleted successfully!"
}

Name: Aditya Wadkar
Roll: 33379
Batch: N-11

Assignment 4(a)

Title: Implementation of jquery mobile

Problem statement:
Create a simple mobile website.

Objective: Apply jquery mobile library conventions & demonstrate a website.

Theory:

What is jquery mobile?

jquery mobile is easiest way to build sites and apps that are accessible on all popular smartphone tablet and desktop.

jquery mobile is a technique optimized HTML5 UI framework designed to make responsive web sites and that are accessible on all smartphone devices.

jquery mobile has multiple pages & navigation widgets, form widgets etc.

① **Pages:**

- The pages is a primary unit of interaction in Jquery mobile & is used to group content into logical views that can be animated in and out of view with page transition.
- A HTML ~~doc~~ document can be built with multiple 'pages' inside it & framework will transition between these local with no need to request content from server

PICT, PUNE

② Widgets:

Navbar:

jQuery Mobile has very basic navbar widget that is useful for providing up to 5 buttons with optional icons in bar.

③ Form Widgets:

CheckBox:

checkBox input are used to provide a list of options where more than one can be selected.

checkbox button are enhanced by checkbox or radio widget

syntax:

```
<label><input type="checkbox" name="checkbox-0">
  check me </label>
```

Radio Buttons:

Radio inputs are used to provide a list of options where only a single option can be selected.

Radio buttons are enhanced by checkbox radio widget.

syntax:-

```
<form>
  <label>
    <input type='radio' name="radio-choice-0">
      One </label>
  <label for="radio-choice-ob"> Two</label>
    <input type="radio" name "radio-choice-0"
    id="radio-choice-ob" class="custom">
</form>
```

Viewport meta tag:

The viewport meta tag is used to specify how the browser should display page. you can set viewport values to disable zooming if required since this is part of your page content, not library.

Multipage template structure:

A single HTML document can contain multiple 'pages' that are loaded together by stacking multiple divs with data-role of page.

* Ajax navigation:

jQuery mobile includes navigation system to load pages into DOM via Ajax, enhance the new content, then display pages with a rich set of animated transitions.

The navigation system uses progressive enchantment to automatically 'hijack' standard links and form submission & route them as a Ajax Request.

One of jQuery mobile core feature is ability to load and view content from a separate pages into initial document with support for standard navigation methods like anchors & back button.

Conclusion:

Thus, we are able to create a simple mobile website using jquery mobile.

20/4

Assignment 4A

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible"
content="IE=edge">
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<link rel="stylesheet"
href="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
1.4.5.min.css" />
<script src="https://code.jquery.com/jquery-
1.11.1.min.js"></script>
<script
src="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
1.4.5.min.js"></script>
<title>Demo of jQuery Mobile</title>
</head>
<body>
<div data-role="page" style="background-
color:aliceblue;">
<div data-role="header" style="background-color:
#2196F3; color: #fff; text-align: center;">
<h1 style="font-size: 24px;">Furniture For You</h1>
</div>
<div data-role="navbar" style="background-color:
#f5f5f5;">
<ul>
<li><a href="index.html" style="color:
#666;">Home</a></li>
<li><a href="about.html" style="color:
#666;">About</a></li>
<li><a href="login_register.html" style="color:
#666;">Register for Newsletter</a></li>
</ul>
</div>
<center>
<h3>Gallery</h3>
</center>
<div class="content">
<center>
<div class="rows">
<img src="img1.jpg" alt="Img1" width="240"
height="200">
<img src="img2.jpg" alt="Img2" width="240"
height="200">
<img src="img3.jpg" alt="Img3" width="240"
height="200">
<img src="img4.jpg" alt="Img4" width="240"
height="200">
<img src="img5.jpg" alt="Img4" width="240"
height="200">
<img src="img6.jpg" alt="Img4" width="240"
height="200">
</div>
<br>
<div class="rows">
<img src="img1.jpg" alt="Img1" width="240"
height="200">
<img src="img5.jpg" alt="Img4" width="240"
height="200">
<img src="img6.jpg" alt="Img4" width="240"
height="200">
</div><br>
</center>
</div>
<div data-role="footer">
<center>
<a href="index.html" class="ui-btn
ui-btn-inline">Home</a>
<a href="about_us.html" class="ui-btn ui-btn-
inline">About</a>
<a href="login_register.html" class="ui-btn ui-btn-
inline">Register</a>
</center>
</div>
</div>
</body>
</html>
```

Register.html

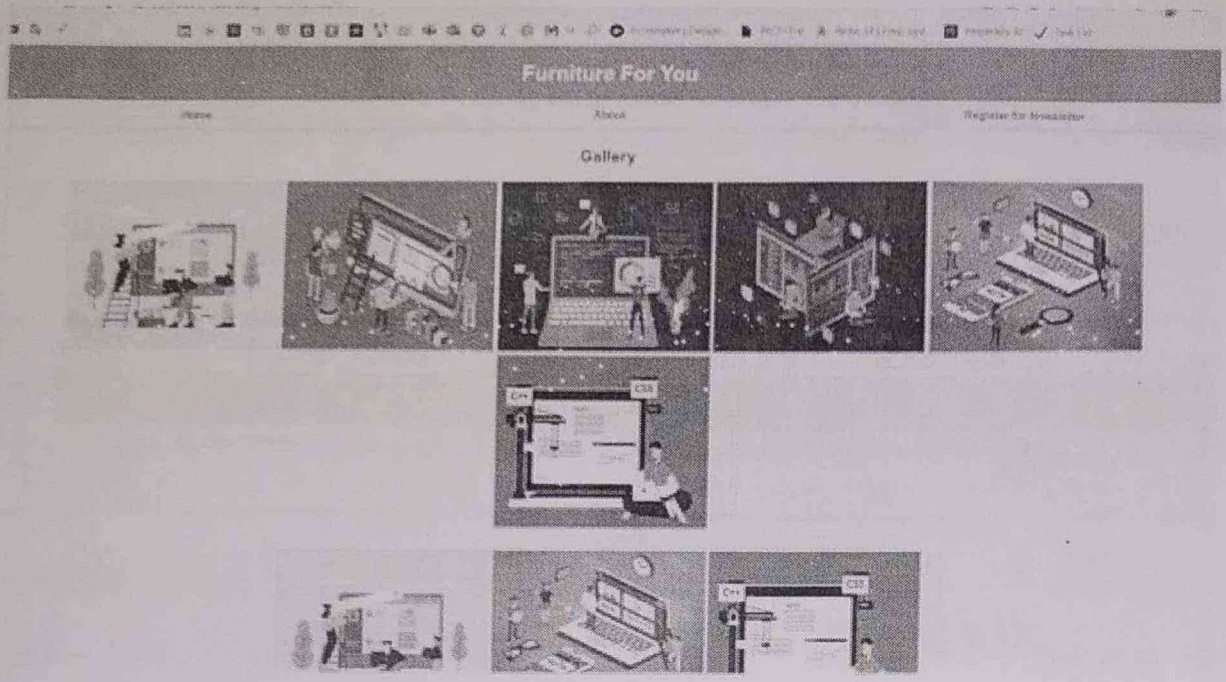
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible"
content="IE=edge">
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<link rel="stylesheet"
href="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
1.4.5.min.css" />
<script src="https://code.jquery.com/jquery-
1.11.1.min.js"></script>
<script
src="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
1.4.5.min.js"></script>
<title>Demo of jQuery Mobile</title>
</head>
<body>
<div data-role="page" style="background-
color:aliceblue;">
<div data-role="header" style="background-color:
#2196F3; color: #fff; text-align: center;">
<h1 style="font-size: 24px;">Furniture For You</h1>
</div>
<div data-role="navbar" style="background-color:
#f5f5f5;">
<ul>
<li><a href="index.html" style="color:
#666;">Home</a></li>
<li><a href="about.html" style="color:
#666;">About</a></li>
<li><a href="login_register.html" style="color:
#666;">Register for Newsletter</a></li>
```

```
</ul>
</div>
<div class="content" style="width:50%; margin: 5%
auto;">
<center><h3>Register</h3></center>
<div data-role="content">
<label for="userName">Username</label>
<input type="text" name="userName">
<label for="email">Email</label>
<input type="text" name="email">
<label for="password">Password</label>
<input type="text" name="password"><center><a
href="index.html" class="ui-btn ui-btn-inline">Register
Me</a></center>
</div>
</div>
</div>
</body>
</html>
```

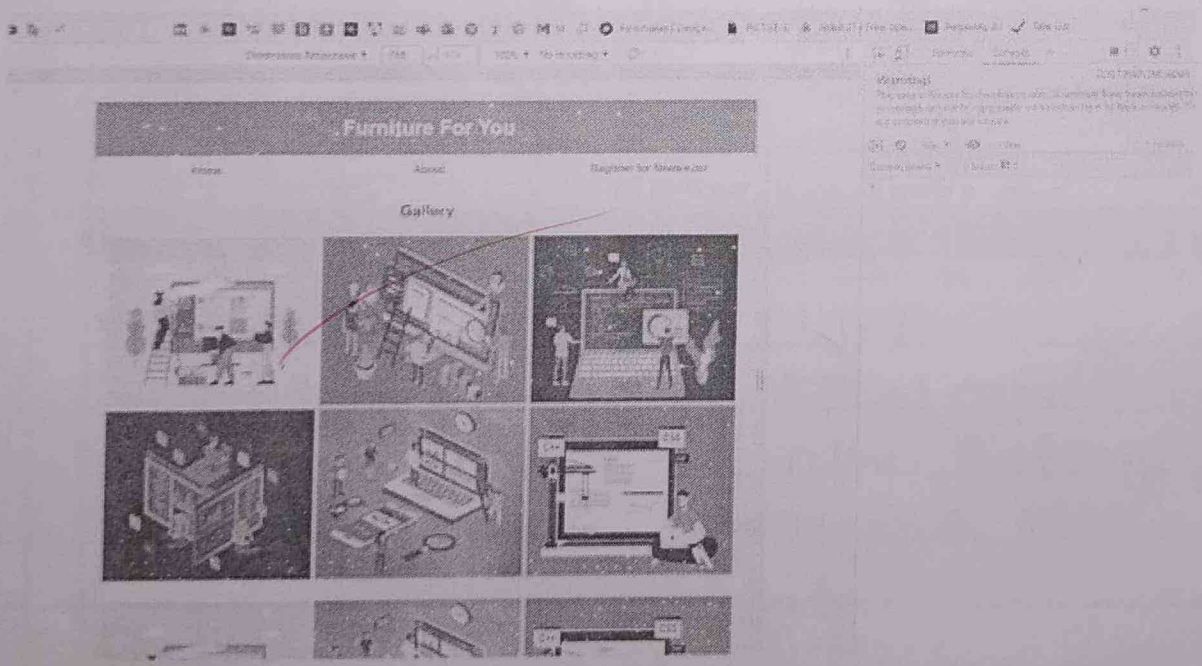
About.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible"
content="IE=edge">
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<link rel="stylesheet"
href="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
1.4.5.min.css" />
<script src="https://code.jquery.com/jquery-
1.11.1.min.js"></script>
<script
src="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
1.4.5.min.js"></script>
<title>Demo of jQuery Mobile</title>
</head>
<body>
<div data-role="page" style="background-
color:aliceblue;">
<div data-role="header" style="background-color:
#2196F3; color: #fff; text-align: center;">
<h1 style="font-size: 24px;">Furniture For You</h1>
</div>
<div data-role="navbar" style="background-color:
#f5f5f5;">
<ul>
<li><a href="index.html" style="color:
#666;">Home</a></li>
<li><a href="about.html" style="color:
#666;">About</a></li>
<li><a href="login_register.html" style="color:
#666;">Register for Newsletter</a></li>
</ul>
</div>
<center>
<h3>My Gallery</h3>
</center>
<div class="content">
<center>
We have a HD image gallery. All the images are taken from
https://unsplash.com/
</center>
</div>
</div>
</body>
</html>
```

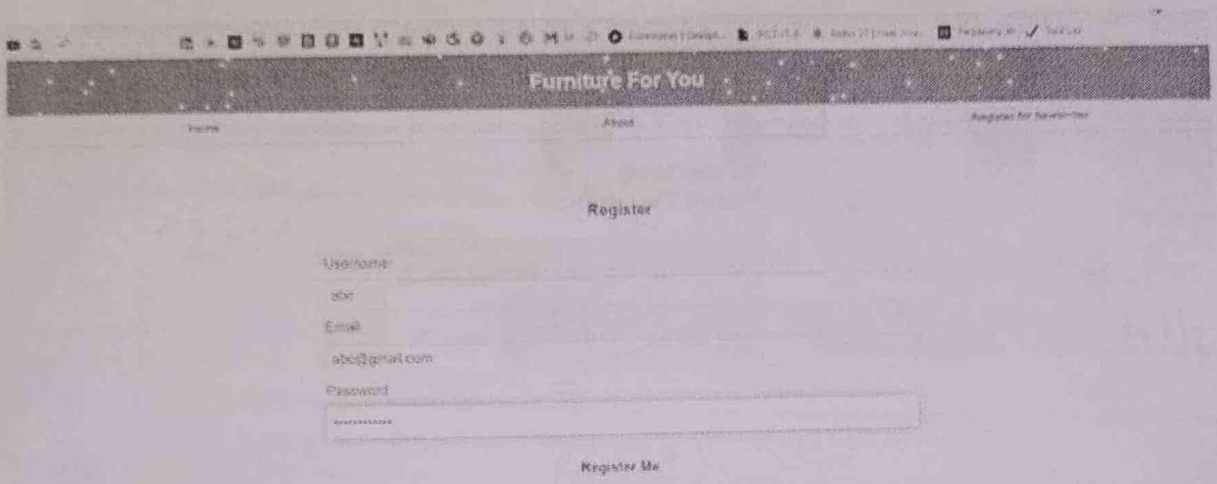
Desktop View



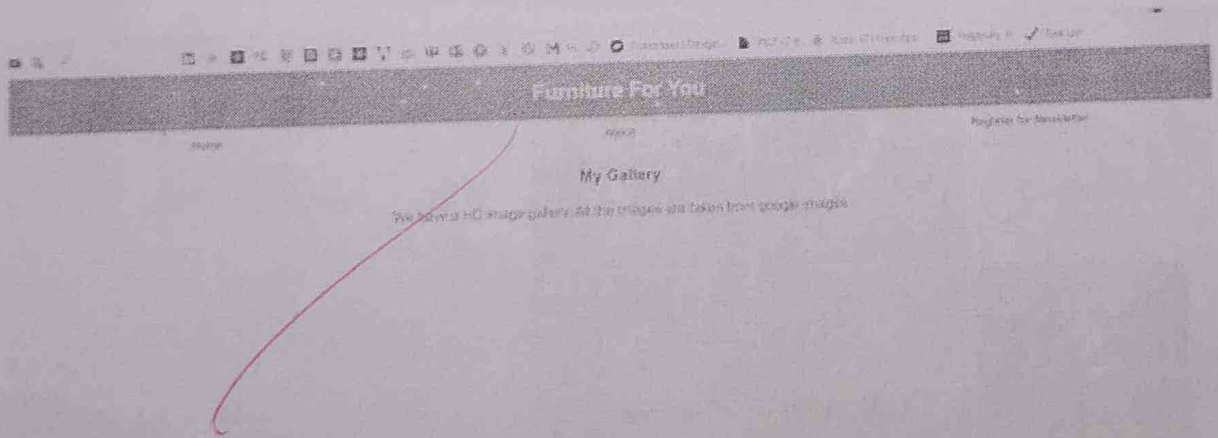
Mobile View



Register.html



About.html



Name : Aditya wadkar
Class : TE I1 Batch : N-11
Roll no : 33379

Assignment 4 (b)

Title : Work on AWS VPC or AWS Elastic Beanstalk

Problem statement : Deploy web application of AWS VPC or AWS Elastic Beanstalk

Objective : understand working of AWS VPC or AWS Elastic Beanstalk

Theory :

What is Amazon VPC

Amazon VPC enables you to launch AWS resources into a virtual n/w that you've defined. This virtual n/w closely resembles a traditional n/w that you'd operate in your own data center.

Features :

The following feature help to configure VPC virtual private clouds (VPC) :

A VPC is a virtual n/w closely resembles a traditional n/w that you'd operate in your own data center.

Subnet :

A subnet is a range of IP addressing in your VPC

IP addressing :
you can assign IPV4 address to VPC

Routing :
user route tables to determine where n/w traffic from your subnet or gateway is directed

Gateway & endpoints:
A gateway connects VPC to another n/w

peering connections:
use a VPC peering connection to route traffic between resources in two VPC's

Traffic mirroring :
copy n/w traffic from n/w interface & send it to security and monitoring appliances for deep packet inspection.

Transit Gateway :
use a transit gateway, which acts as a central hub, to route traffic between your VPC's, VPN connection and AWS

VPC flow log :
A flow log captures information about IP traffic going to and from n/w interfaces.

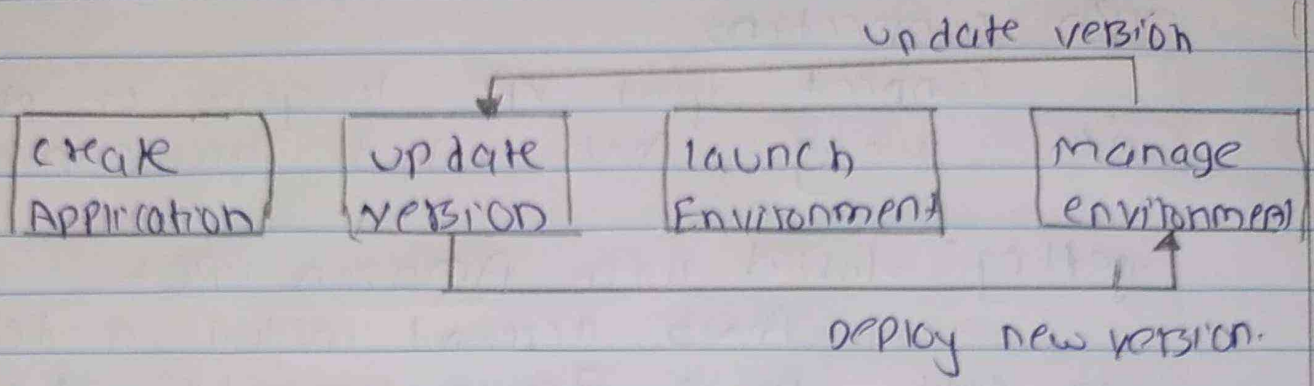
VPN connections:
connect your VPC to your on-premises n/w using AWS virtual private n/w.

getting started with Amazon VPC:
your AWS account includes a default VPC in each AWS Region. your default VPC are configured such that you can immediately start launching & connection to EC2 instance

* working with Amazon VPC:
You can create, manage your VPC using:
- AWS management console
- AWS CLI
- AWS SDK
- Query API

How the Amazon VPC works:
Amazon VPC enables to launch AWS resources into virtual n/w that you've defined

what is AWS Elastic Beanstalk?
AWS comprises over one hundred services each of which express an area of functionality while variety of services offer flexibility for how you want to manage your infrastructure.



* Setting up → create Aws account

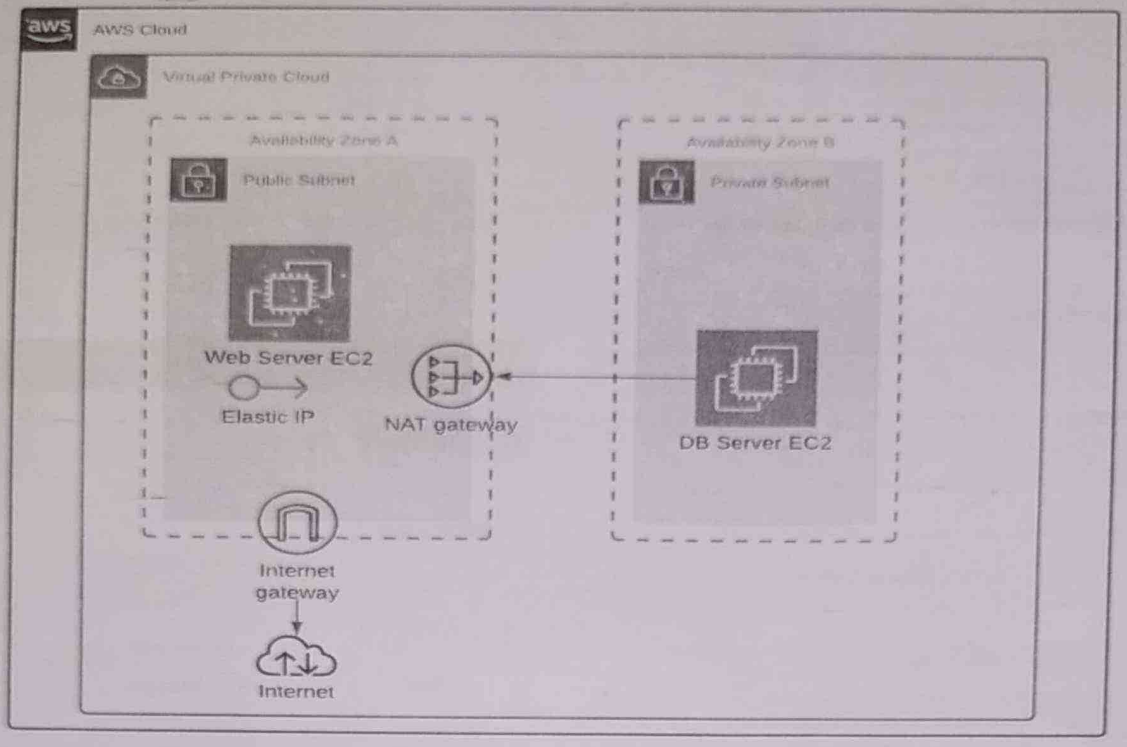
① create an example Application
② Explore your environment
③ deploy new version of your application
④ configure your environment
⑤ clean up.

Conclusion :-

Thus, we studied how to deploy / host web application on Aws VPC or Aws elastic Beanstalk.

Assignment -4b

Title: Work on AWS VPC or Elastic Beanstalk for Deployment.



Create VPC

A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances.

VPC settings

Resources to create

VPC only

VPC and more

Name tag - optional

MyVPC

IPv4 CIDR block

IPv4 CIDR manual input

IPAM-allocated IPv4 CIDR block

IPv4 CIDR

10.0.0.0/16

IPv6 CIDR block

No IPv6 CIDR block

IPAM-allocated IPv6 CIDR block

Amazon-provided IPv6 CIDR block

IPv6 CIDR owned by me

Tenancy

Default

You successfully created vpc-055ba63a2afc5eb5f / MyVPC

VPC > Your VPCs > vpc-055ba63a2afc5eb5f

vpc-055ba63a2afc5eb5f / MyVPC

Details

VPC ID

vpc-055ba63a2afc5eb5f

State

Available

Tenancy

Default

DHCP option set

Default VPC

No

IPv4 CIDR

10.0.0.0/16

Route 53 Resolver DNS Firewall rule groups

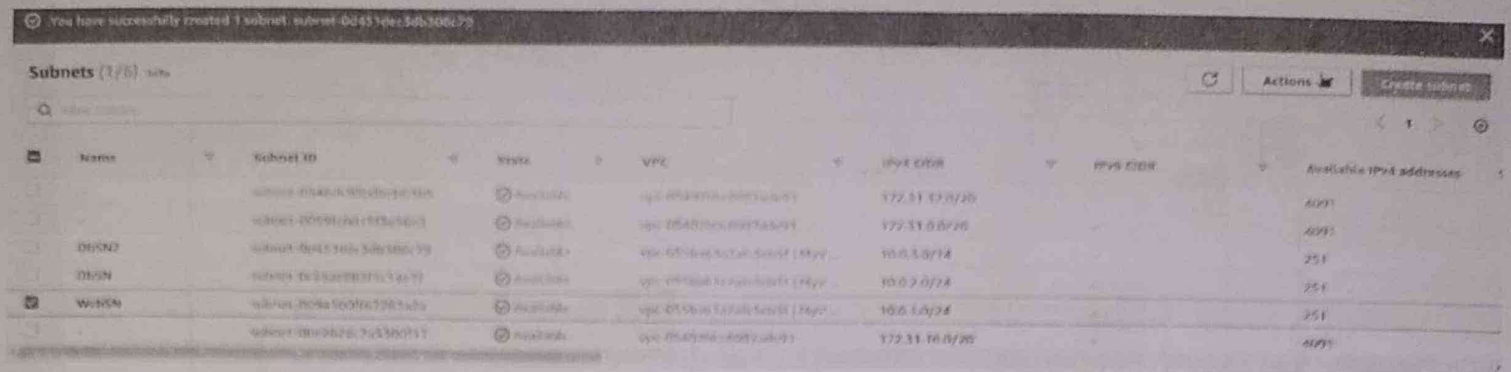
Owner ID

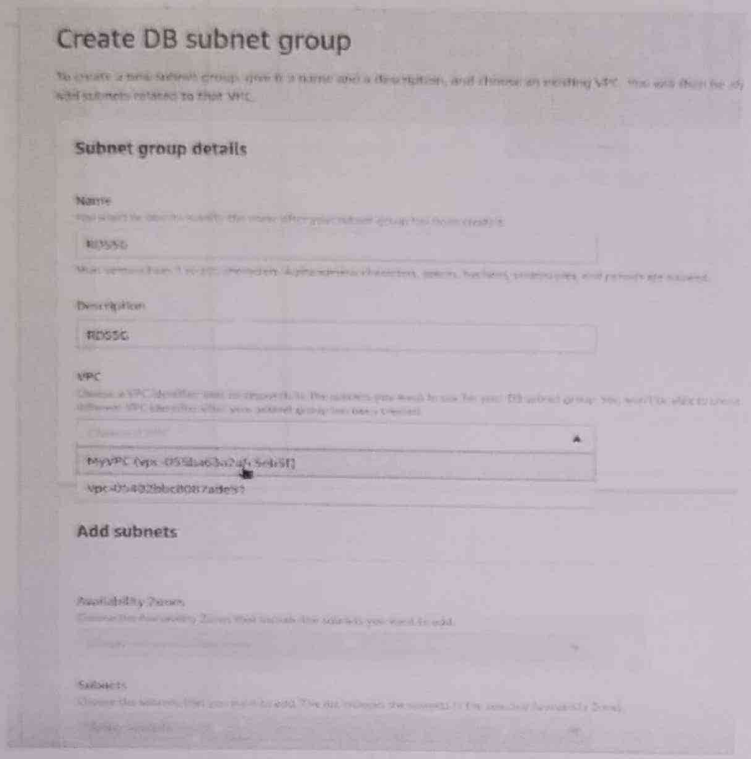
102136579241

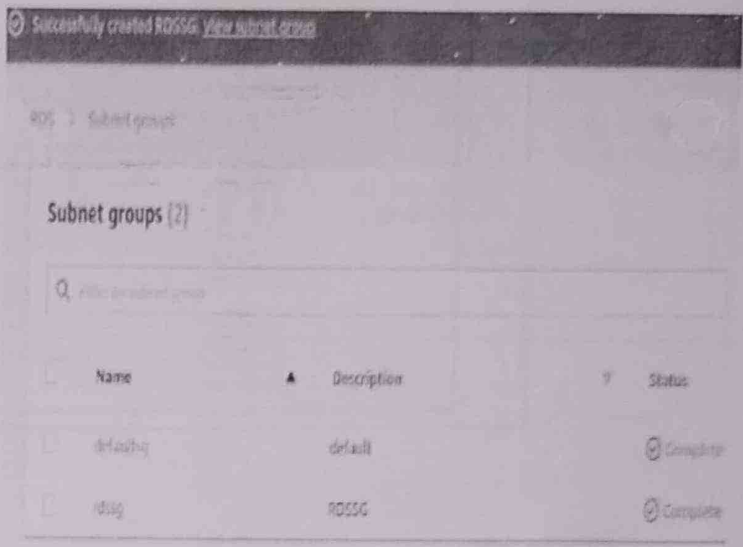
CIDRs | Flow logs | Tags

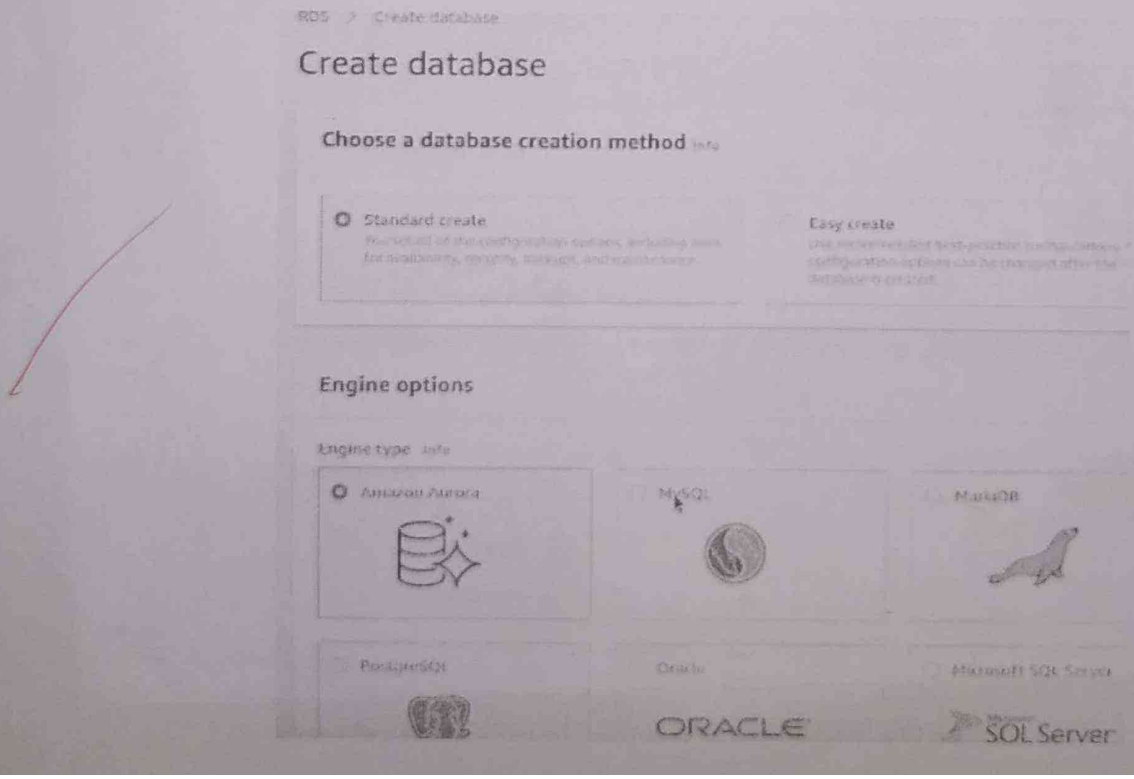
CIDRs

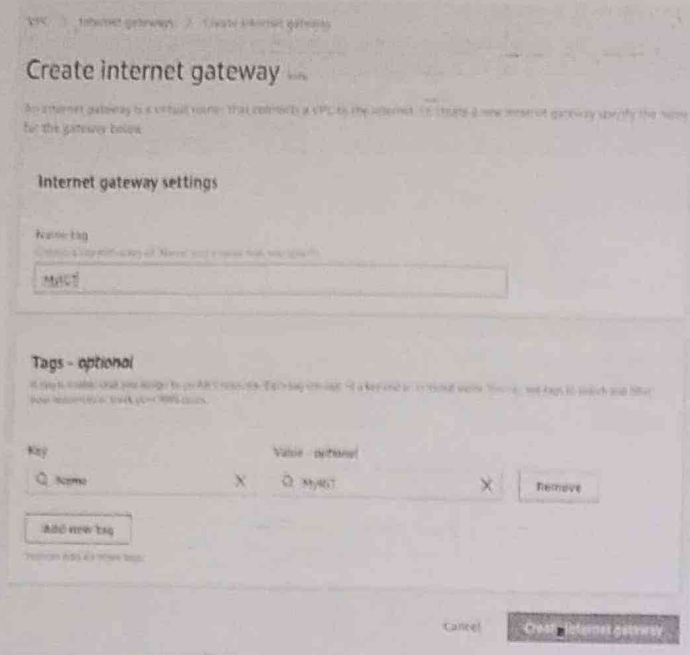
Address type	CIDR
IPv4	10.0.0.0/16

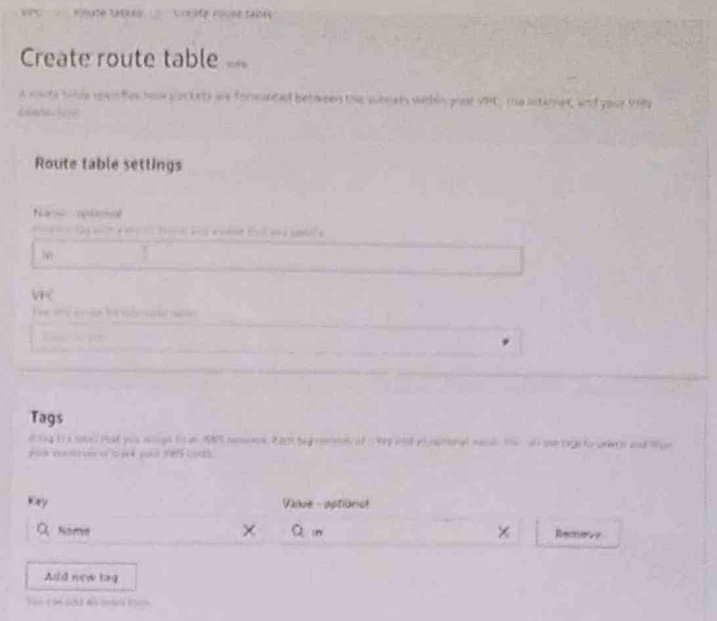

You have successfully created 1 subnet: subnet-0d451de3db3086278
Subnets (1/6) Info
Actions
Create subnet
Name
Subnet ID
State
VPC
IPv4 CIDR
IPv6 CIDR
Available IPv4 addresses
DBSN2
DBSN
WebSN

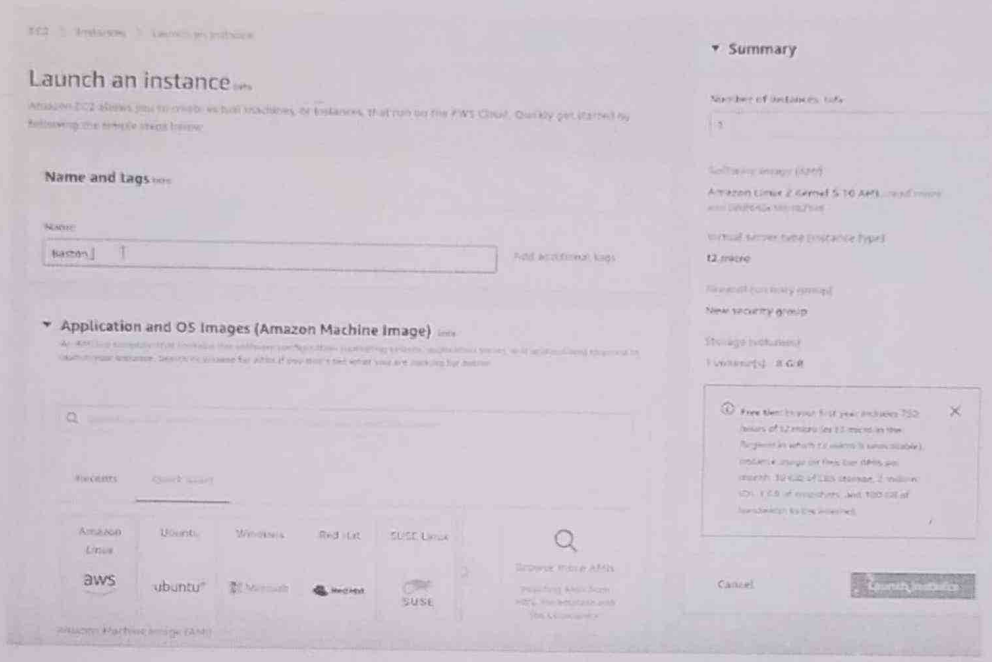

Create DB subnet group
Subnet group details
Name
RDSSG
Description
RDSSG
VPC
MyVPC (vpc-0558ae63a2af1e4e5f)
Add subnets
Availability Zones
Subnets

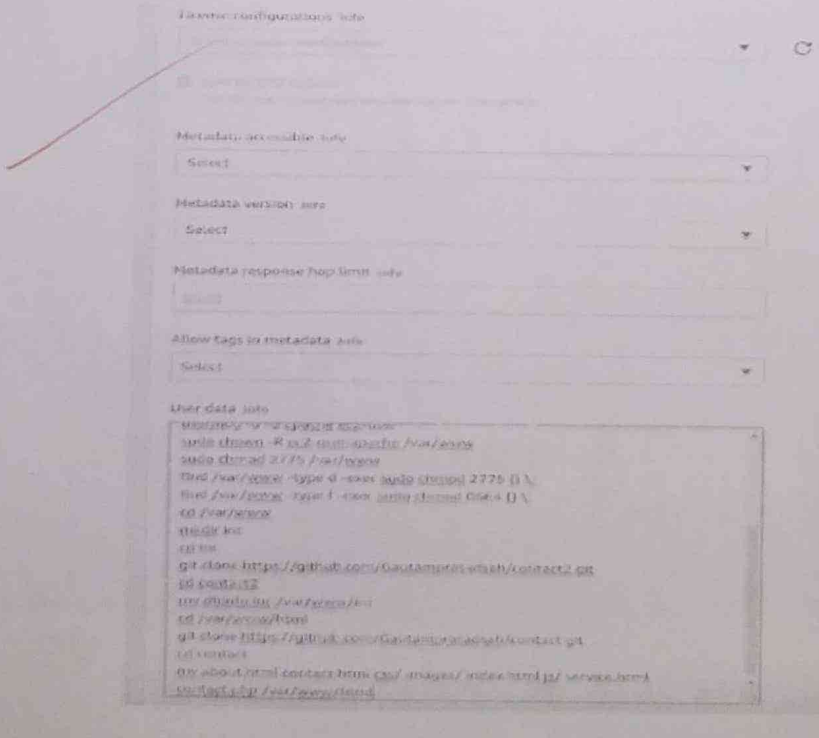

Successfully created RDSSG. View subnet group
RDS > Subnet groups
Subnet groups (2)
Name
Description
Status
default
RDSSG

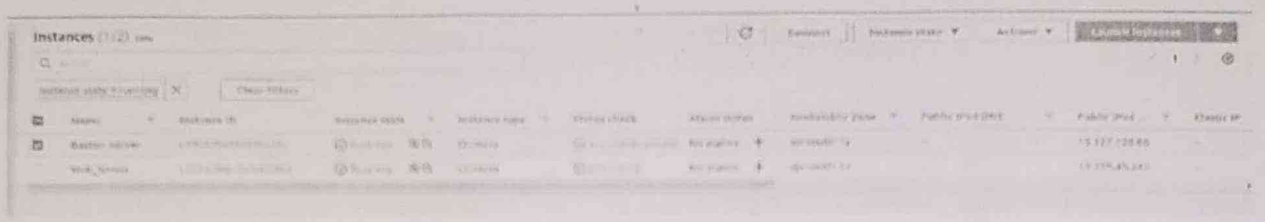

RDS > Create database
Create database
Choose a database creation method Info
Standard create
Easy create
Engine options
Engine type Info
Amazon Aurora
MySQL
MariaDB
PostgreSQL
Oracle
Microsoft SQL Server
ORACLE
SQL Server

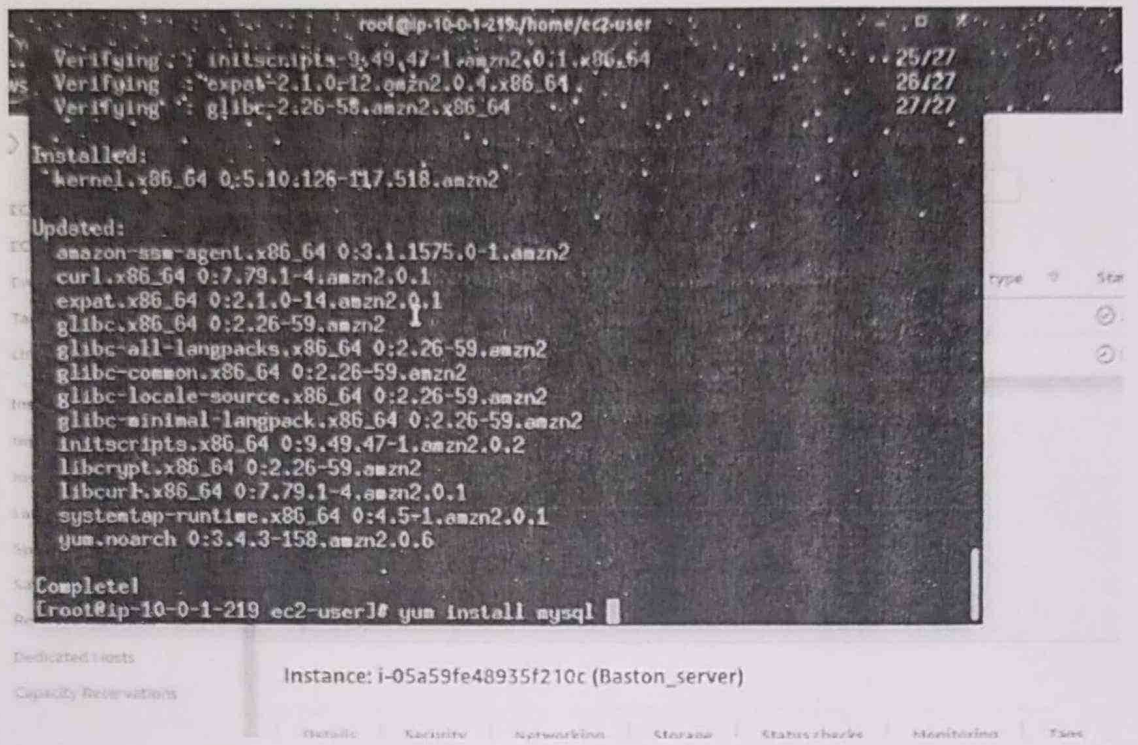

Create internet gateway
Internet gateway settings
Name tag
MyIGT
Tags - optional
Key
Value - optional
Name
MyIGT
Remove
Add new tag
Cancel

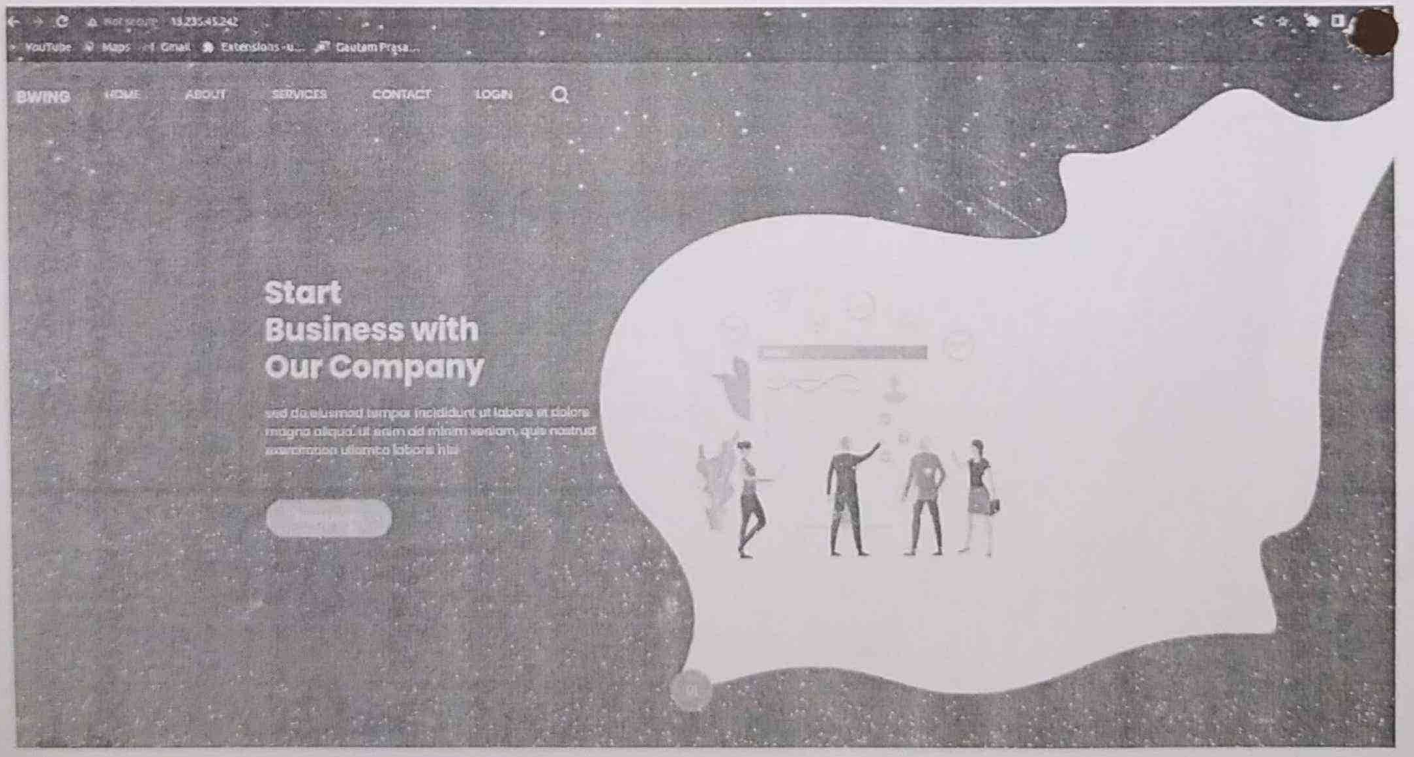

Create route table
Route table settings
Name - optional
VPC
Tags
Key
Value - optional
Name
in
Remove
Add new tag


Launch an instance
Name and tags
Name
Add additional tags
Application and OS Images (Amazon Machine Image)
Summary
Number of instances
Software Image (AMI)
Amazon Linux 2 kernel 5.10 AMI
Virtual server type (instance type)
t2.micro
Firewall (security group)
New security group
Storage (volumes)
Cancel


Metadata accessible
Select
Metadata version
Select
Metadata response hop limit
Allow tags in metadata
Select
User data
sudo chown -R ec2-user:apache /var/www
sudo chmod 2775 /var/www
find /var/www -type d -exec sudo chmod 2775 {} \;
find /var/www -type f -exec sudo chmod 0664 {} \;
cd /var/www
mkdir inc
cd inc
git clone https://github.com/Gautamprats-wash/Contact2.git
cd Contact2
mv images/* /var/www/inc
cd /var/www/html
git clone https://github.com/Gautamprats-wash/Contact.git
cd Contact
mv about.html contact.html css/ images/ index.html js/ service.html
/var/www/html




root@ip-10-0-1-219:/home/ec2-user
Verifying : initscripts-9.49.47-1.amzn2.0.1.x86_64 25/27
Verifying : expat-2.1.0-12.amzn2.0.4.x86_64 26/27
Verifying : glibc-2.26-58.amzn2.x86_64 27/27
Installed:
kernel.x86_64 0:5.10.126-117.518.amzn2
Updated:
amazon-ssm-agent.x86_64 0:3.1.1575.0-1.amzn2
curl.x86_64 0:7.79.1-4.amzn2.0.1
expat.x86_64 0:2.1.0-14.amzn2.0.1
glibc.x86_64 0:2.26-59.amzn2
glibc-all-langpacks.x86_64 0:2.26-59.amzn2
glibc-common.x86_64 0:2.26-59.amzn2
glibc-locale-source.x86_64 0:2.26-59.amzn2
glibc-minimal-langpack.x86_64 0:2.26-59.amzn2
initscripts.x86_64 0:9.49.47-1.amzn2.0.2
libcrypt.x86_64 0:2.26-59.amzn2
libcurl.x86_64 0:7.79.1-4.amzn2.0.1
systemtap-runtime.x86_64 0:4.5-1.amzn2.0.1
yum.noarch 0:3.4.3-158.amzn2.0.6
Complete!
[root@ip-10-0-1-219 ec2-user]# yum install mysql
Instance: i-05a59fe48935f210c (Baston_server)


BWING
HOME
ABOUT
SERVICES
CONTACT
LOGIN
Start
Business with
Our Company