A Project Report

on

Ecommerce Customer Side Application

Submitted under skill-oriented course (MSD)

Under the Faculty Guideship of

Mr. K. Pradeep

Asst. Professor

Vignan's Lara Institute of Technology & Science

Submitted by:

PANDI VENKATESWARLU

22FE1A05C5

APRIL - 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project report entitled **Ecommerce Customer side Application** is a Bonafide work carried out by **PANDI VENKATESWARLU, 22FE1A05C5**, under the guidance of **Mr. K. Pradeep, Assistant Professor** in skill-oriented course (MEAN Stack Technologies) and submitted in fulfilment of the requirements for the award of the degree of Bachelor of Technology in **COMPUTER SCIENCE AND ENGINEERING** from **VIGNANS LARA INSTITUTE OF TECHNOLOGY AND SCIENCES, VADLAMUDI**, during the academic year 2024-25.

Project Guide	**Head of the Department**
Mr. K. Pradeep	**Dr. K. Venkateswara Rao**
Assistant Professor	**Professor**

External Examiner

STUDENT'S DECLARATION

I PANDI VENKATESWARLU, a student of Bachelor of Technology Reg. No: **22FE1A05C5** of Department of Computer Science and Engineering. I do here by declare that I have completed the **FULL STACK PROJECT** under skill oriented in my 3-2 semester to enhance technical skills and gained hands-on experience in WEB DEVELOPMENT under the faculty guidance of Mr. **K. PRADEEP,** Department of Computer Science and Engineering in college.

(Signature and Date)

CERTIFICATE

udemy

Certificate no: UC-ce6cba78-6d1f-4666-b67b-bc3bfe223a61
Certificate url: ude.my/UC-ce6cba78-6d1f-4666-b67b-bc3bfe223a61
Reference Number: 0004

CERTIFICATE OF COMPLETION

The Complete Full-Stack Web Development Bootcamp

Instructors **Dr. Angela Yu, Developer and Lead Instructor**

Pandi Venkateswarlu

Date **April 21, 2025**
Length **61.5 total hours**

ABSTRACT

This project presents the development of a **Customer-Side E-Commerce Web Application** designed using the **MEAN Stack** technologies — **MongoDB, Express.js, Angular (or EJS for templating), and Node.js**. The primary objective of the application is to simulate a real-time shopping experience for users, allowing them to explore various product categories, manage their accounts, and perform essential e-commerce operations such as searching products, adding items to a cart, and viewing the total cost of selected products.

The project emphasizes **user authentication and session management**, enabling users to register, log in, reset passwords, and maintain their cart across browsing sessions. Products are categorized under sections like **Mobiles & Accessories**, **Clothing**, and **Medicines**, each presented with images, descriptions, and prices. Users can dynamically search categories, seamlessly add or remove items from their cart.

All data is stored and managed using **MongoDB**, with Mongoose. The backend is powered by Express.js and Node.js, ensuring smooth server-side logic and secure session handling. The UI is crafted using EJS and styled with modern CSS for an intuitive and responsive design.

This project not only demonstrates practical implementation of full-stack web development skills but also mimics real-world e-commerce functionalities, making it a robust, user-friendly, and scalable solution suitable for further enhancement into a production-level platform.

INDEX

I.College Certificate	**3**
II.Student's Declaration	**4**
III.Acknowledgement	**5**
IV.Course Certificate	**6**
V.Abstract	**7-12**
VI.Index	**13**
VII.Introduction	**14**
VIII.Objectives	**15**
IX.Hardware and Software Requirements	**16**
X.Modules Description	**17-26**
XI.Capstone Project 1	**27-29**
XII.Capstone Project 2	**30-34**
XIII.Capstone Project 3	**35-43**
XIV.Capstone Project 4	**44-47**
XV. Capstone Project 5	**48-55**

INTRODUCTION

Electronic Commerce, commonly known as **E-Commerce,** is a wide industry, which refers to buying and selling of the goods and services, over the internet. It has a wide range of online business activities, including retail, wholesale services. This E-Commerce provides convenience to customers, allowing them to shop their specific products anytime and anywhere. It offers the businesses a wide reach, enabling them to access the global markets and reduce the operational costs.

This E-Commerce industry have segmented into various categories like as follows

1. Business to Customer
2. Business to Business
3. Customer to Business
4. Customer to Customer

The E-Commerce industry experiencing a significant growth over the past few decades, increases internet penetration, and changing customer behaviors. The present cutting-edge technology **artificial intelligence (AI)**, was integrated in this E-Commerce, which helps the users to personalize their shopping in an easy way through world-wide.

My E-Commerce application is a customer-side application, where the customers or users of my application can able to shop and buy their required items in anywhere at any time. In my application, the user can create an account if it doesn't exist or can able to login through their valid credentials. Later, the dashboard will be displayed to the user, where the user can able to search for the items, view his/her item's cart, and logout if their shopping was over. After checking the cart, the user will able to pay for what he/she wants to buy. The users can able to delete the items if they don't need.

OBJECTIVES

The objectives of my E-Commerce customer side application are:

- **Enhance User experience**
  - My E-Commerce customer side application have an user-friendly interface that simplifies the shopping experience for the users.
  - It implements the features like improve navigation, product search and overall usability of the platform.
- **Implement robust security measures**
  - This application establishes the strong security protocols to protect the customer data.
  - It complies the industry standards regarding the data privacy and security.
- **Promote Sustainability**
  - It incorporates the eco-friendly practices in the product offerings.
  - It raises awareness on the sustainable shopping choices among the customers
- **Expand the market reach**
  - It is a scalable platform that can accommodate and expand the growth into the new markets.
  - It implements the market strategies that target diverse customer segments and geographical segmentations.
- **Facilitate Inventory management**
  - It consists of a robust backend system, that allows the users for easy product management, its listing, and order fulfilment.

HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements:

- Inter I5 or I7 Processor
- 8GB RAM
- 512GB SSD/HDD
- Stable Internet/LAN Connection

Software Requirements:

- Visual Studio Code (VS Code)
- Node JS

A Browser (Google Chrome or Microsoft Edge or Firefox)

MODULES DESCRIPTION

1. User Authentication Module

Purpose: Handles user registration, login, and session management.

Key Files:

- register.ejs
- login.ejs
- server.js

Functionalities:

- Registration via /register → Stores user details in MongoDB (userbase collection).
- Login via /verify:
  - Validates credentials.
  - Creates a session (req.session.user) to track logged-in users.
- Password match is checked at registration.
- Session is required for protected actions (like enrolment).

2. Dashboard Module

Purpose: Views the user's dashboard, and user can browse their specific items

Key Files:

- dashboard.ejs
- category.ejs
- viewcart.ejs
- cart.js

Functionalities:

- User can browse the items of a specific category
- User can view his/her cart of added items
- And also, user can logout from his account, when he/she done the shopping. In the logout process, the created session will be disabled.

3. Categories Module

Purpose: User can search for specific category or cluster of items, and add them to the cart

Key Files:

- category.ejs
- mobiles.ejs
- clothes.ejs
- medicines.ejs
- cart.ejs
- cart.js

Functionalities:

- user can select the category whatever he wants.
- User can browse for specific item he wants
- User can add the specific item he wants in the cart, and can view his cart

4. Cart Module

Purpose: View of cart, and confirming the cart **Key**

Files:

- cart.js

- cart.ejs
- viewcart.ejs

Functionalities:

- The user can able to see his cart
- The user can able to remove the specific items in his cart
- The user can able to perform pay transaction after finalizing the cart.

server.js const exp = require("express"); const bp =

require("body-parser"); const db = require("mongoose");

const app = exp(); const categoryRoutes =

require('./routes/categories'); const Cart =

require('./models/Cart'); const path = require("path");

const session = require("expresssession");

//setup ejs app.set("view engine", "ejs");

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

app.use(exp.static("public")); app.use('/category', categoryRoutes);

app.use(session({ secret:

'secretKeyForSession123', // use env var in production

resave: false, saveUninitialized: true, cookie: { secure:

false } // secure: true if using HTTPS

```
}));

//ensuring the middleware app.use(exp.urlencoded({extended:
true})); app.use(exp.json());

//setup mongodb
db.connect("mongodb://localhost:27017/ECommerce", {
useNewUrlParser: true,     useUnifiedTopology: true
}).then(() => {     console.log("Connected successfully to
MongoDB");
}).catch(err => {
console.error("Connection error:", err);
});

const userSchema = new db.Schema({
    name: String,
username: String,
password: String,     country:
String,     state:
String,     city: String,     contactnumber:
String,
```

```
}, { collection: "users" }); const user =

db.model("user", userSchema);



//localhost

app.get("/",(req,res)=>{

res.render("home");

})



//login form app.get("/login", (req, res) => {

res.render("login", { errorMessage: null });

}); app.post("/login", async (req, res) => {

const { username, password } = req.body;

    try {          const foundUser = await user.findOne({

username });



       if (foundUser && foundUser.password === password) {

req.session.user = foundUser;             res.redirect("/dashboard");

//    CORRECT

       } else {             res.render("login", { errorMessage: "Invalid username or

password" });

       }
```

```
    } catch (err) {         console.error(err);
res.status(500).send("Server error");
    }
});

//dashboard app.get("/dashboard",
(req, res) => {
if(!req.session.user){         return res.redirect("/login");
    }
    res.render("dashboard",{user: req.session.user});
});

//registration form app.get("/register",(req,res)=>{
res.render("register");
}); app.post("/register", async (req, res) => {     const { username, password,
confirmpassword, country, state, city, name, contactnumber } = req.body;
    try {         let
insertUser;

        if (password === confirmpassword) {
const newUser = new user({                 username,
password,
```

```
            confirmpassword,
            country,          state,          city,
name,          contactnumber:
String(contactnumber),
        });


        insertUser = await newUser.save();
    }
     if (insertUser) {
return res.redirect("login");
    } else {
        res.send("<h1>Invalid username or password</h1>");
    }
  } catch (err) {      console.error(err);
res.status(500).send("Server error");
  }
});

//forgot password app.get("/forgotPassword",
(req, res) => {   res.render("forgotPassword",
{ errorMessage1: null });

});
```

```
app.post("/forgotPassword", async (req, res) => {     const {
username, new_password, confirm_password } = req.body;
    try {
       if (new_password !== confirm_password) {             return
res.render("forgotPassword", { errorMessage1: "Passwords do not match" });
       }

       const result = await user.updateOne(
          { username: username },
          { $set: { password: new_password } }
       );

       if (result.modifiedCount > 0) {
res.redirect("/login");
       } else {
          res.render("forgotPassword", { errorMessage1: "User not found or password
unchanged" });
       }
   } catch (err) {         console.error(err);
res.status(500).send("Server error"); }
```

```
});

app.get('/search', (req, res) => {     const category =
req.query.category.toLowerCase().trim();

    if (category.includes('mobile')) {
return res.redirect('/category/mobiles');     }
else if (category.includes('cloth')) {
return res.redirect('/category/clothes');     }
else if (category.includes('medicine')) {
return res.redirect('/category/medicines');
} else {         return res.send('Category not
found.');
    }
});

//add to cart functionality app.post('/add-to-cart',
async (req, res) => {     const { productName, price
} = req.body;

    if (!req.session.user) {
   return res.redirect('/login');
```

```
    }     try {        const
cartItem = new Cart({
productName,
          price,
          username: req.session.user.username // associate item with user          });


        await cartItem.save();


        res.send(`
          <h2>Product "${productName}" added to cart successfully!   </h2>
          <a href="/dashboard"><← Back to Dashboard</a>
        `);
    } catch (error) {         console.error(error);
res.status(500).send('Error adding to cart');
    }
});

//viewing cart and total price
app.get('/viewcart', async (req, res) => {  if
(!req.session.user) {     return
res.redirect('/login');
```

```
    }    try {        const userCart = await Cart.find({
username: req.session.user.username });


        res.render('viewCart', { cartItems: userCart, user: req.session.user });
    } catch (err) {        console.error(err);        res.status(500).send("Error
fetching cart items");
    } }); app.get('/cart', async (req,
res) => {    if (!req.session.user) {
return res.redirect('/login');
    }
    try {        const userCart = await Cart.find({ username:
req.session.user.username });


        // Calculate total price        const totalPrice = userCart.reduce((sum,
item) => sum + item.price, 0);


        res.render('cart', { cartItems: userCart, totalPrice });
    } catch (err) {    console.error(err);
res.status(500).send('Error fetching
cart');
    }
```

```
});

//deleting the cart items app.post('/cart/delete/:id',
async (req, res) => {     if (!req.session.user) return
res.redirect('/login');

      try {        await
Cart.findByIdAndDelete(req.params.id);
res.redirect('/cart'); // reload cart after deletion
} catch (err) {       console.error(err);
res.status(500).send("Error deleting item");
   } }); app.get('/cart', async (req, res) => {     if
(!req.session.user) return res.redirect('/login');
      try   {                    const   cartItems   =   await   Cart.find({   username:
req.session.user.username });    res.render('cart', { cartItems });
   } catch (err) {   console.error(err);
res.status(500).send("Error loading
cart");
   }
});

//logout app.get('/logout', (req,
res) => {
```

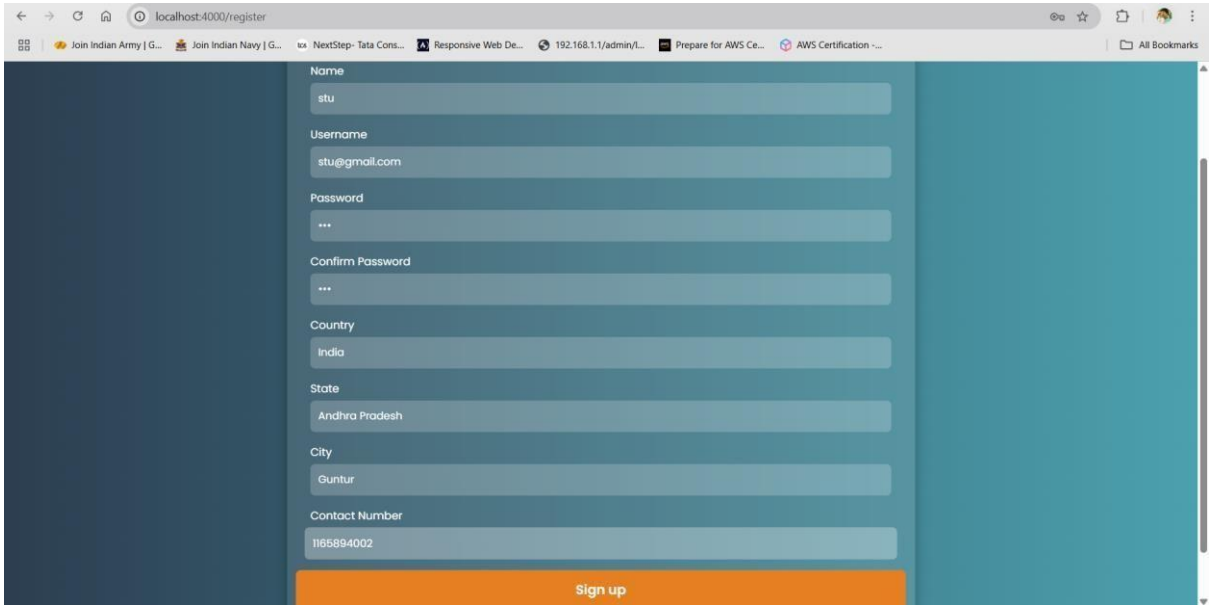
```
req.session.destroy(err => {
if (err) console.error(err);
res.redirect('/login');

    });
});
```

```
//server app.listen(4000,()=>{     console.log("Server is
running at port number 4000")
})
```

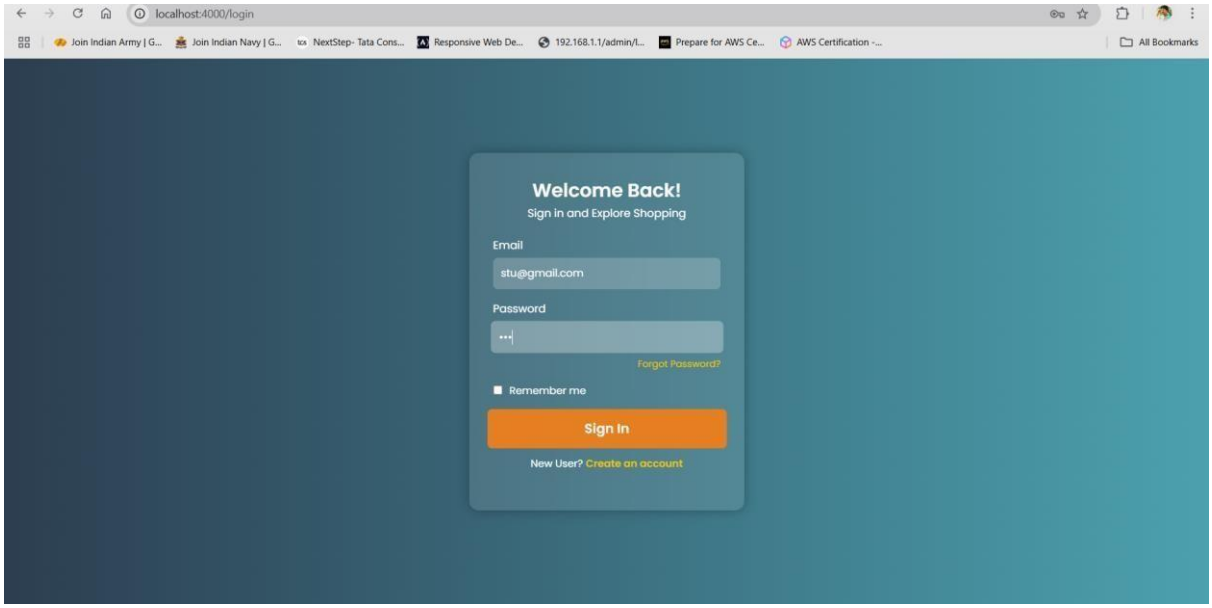
Module Screenshots:

Home page

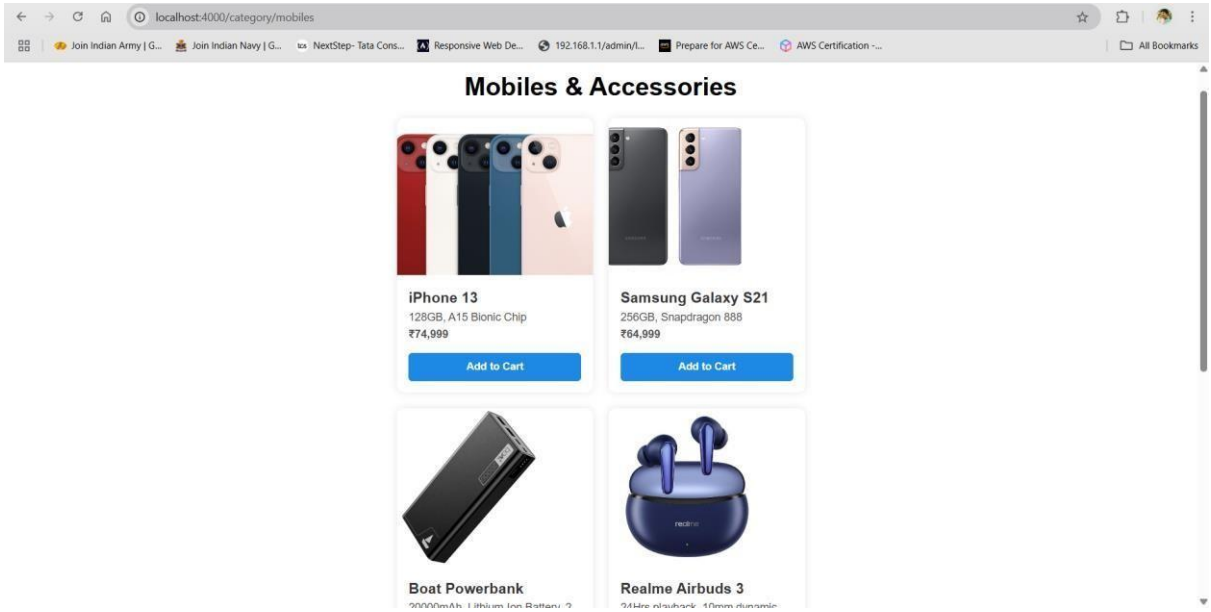
Register page



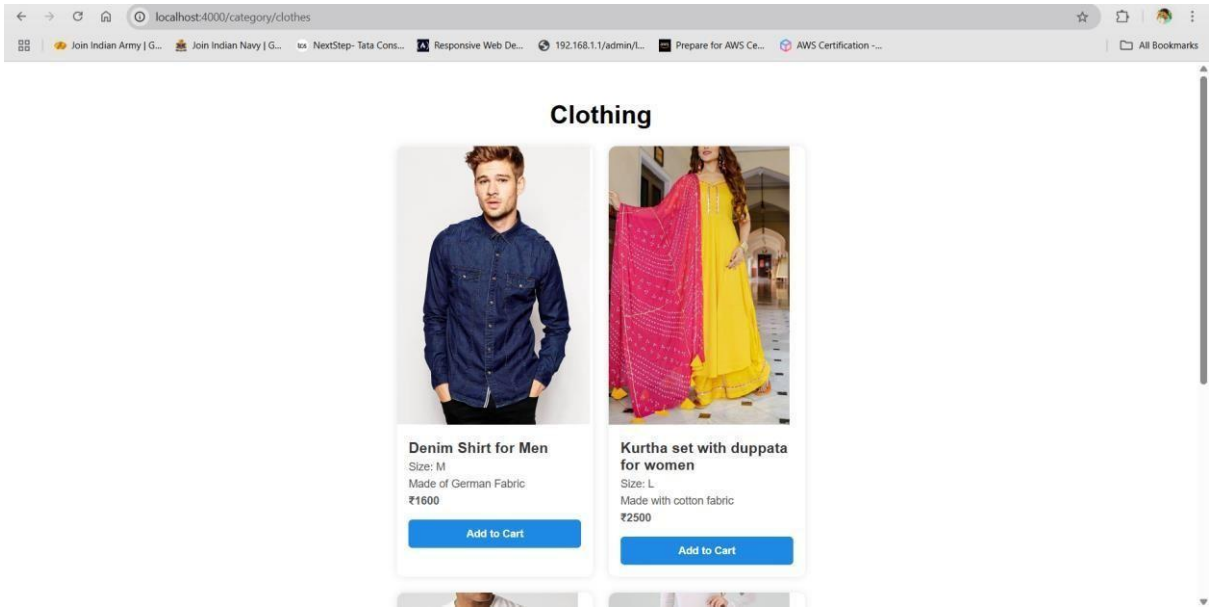
Login page



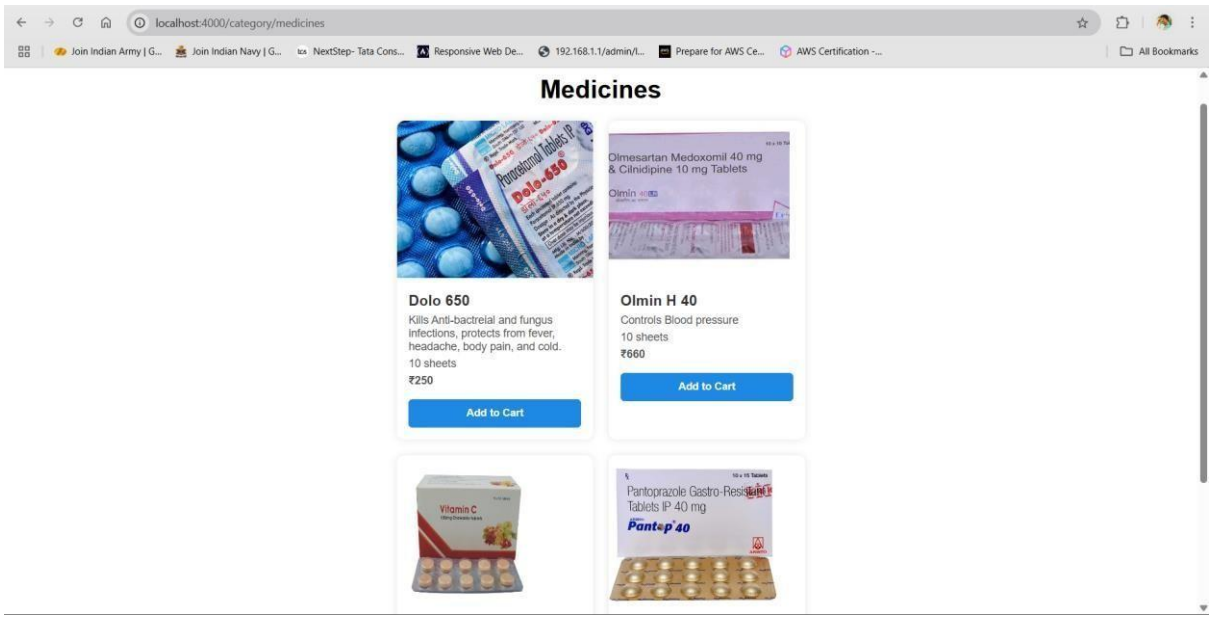
Mobiles and Accessories



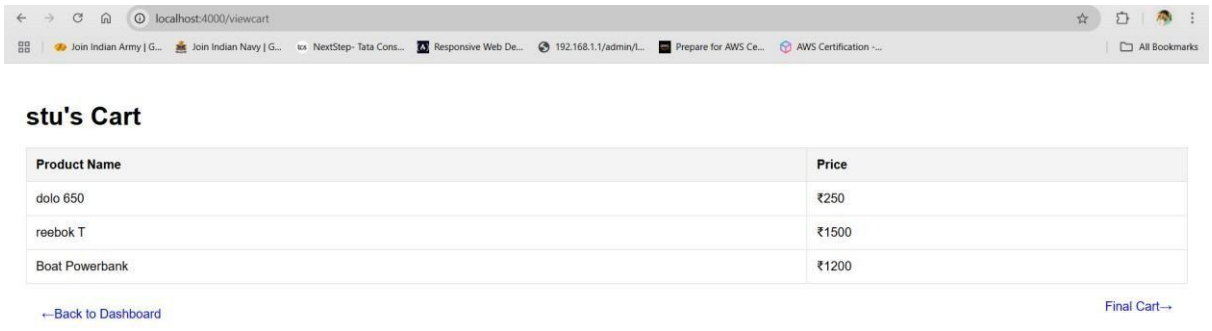
Clothes



Medicines



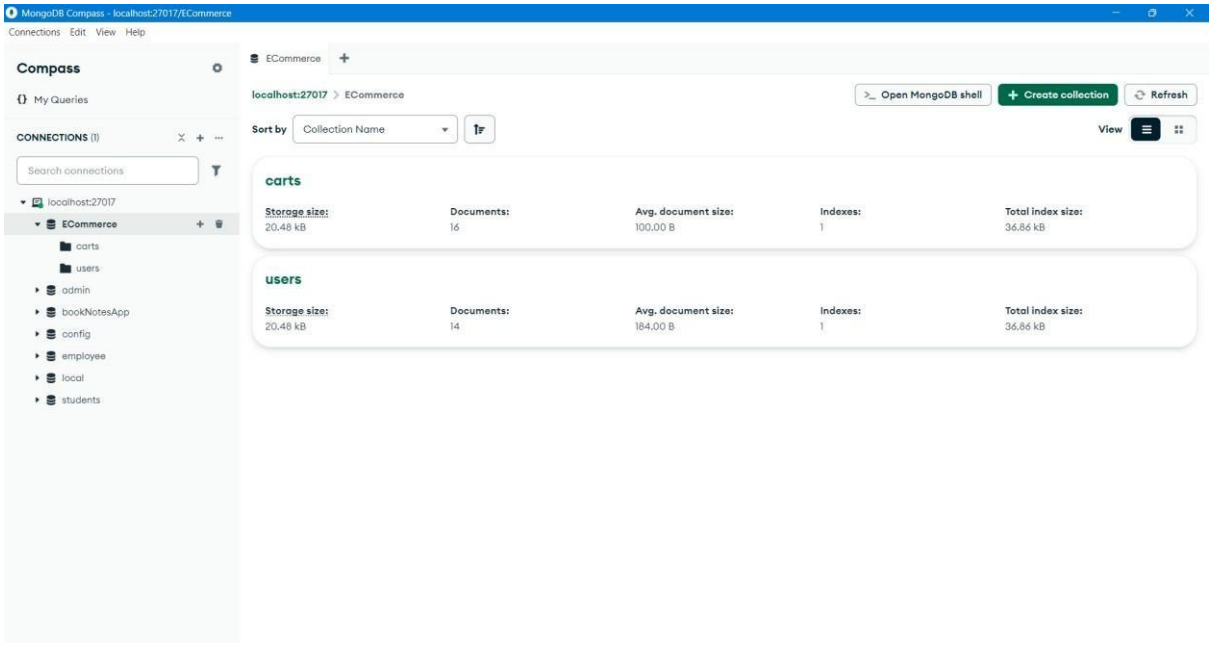
View Cart



Final Cart


localhost:4000/cart
Join Indian Army | G...
Join Indian Navy | G...
NextStep- Tata Cons...
Responsive Web De...
192.168.1.1/admin/l...
Prepare for AWS Ce...
AWS Certification -...
All Bookmarks
Hello, stu@gmail.com!
Your Cart Items:
Product
Price
Action
dolo 650
₹250
Remove
reebok T
₹1500
Remove
Boat Powerbank
₹1200
Remove
Total: ₹2950
←Back to Dashboard
Confirm Pay→

Database


MongoDB Compass - localhost:27017/ECommerce
Connections Edit View Help
Compass
My Queries
CONNECTIONS (1)
Search connections
localhost:27017
ECommerce
carts
users
admin
bookNotesApp
config
employee
local
students
ECommerce
localhost:27017 > ECommerce
Open MongoDB shell
Create collection
Refresh
Sort by
Collection Name
View
carts
Storage size:
20.48 kB
Documents:
16
Avg. document size:
100.00 B
Indexes:
1
Total index size:
36.86 kB
users
Storage size:
20.48 kB
Documents:
14
Avg. document size:
184.00 B
Indexes:
1
Total index size:
36.86 kB

CAPSTONE PROJECT 1

ONLINE RESUME

Objective Of Activity Done: Introduction of HTML, preparation of sample resume

Detailed Report:

Before implementation of this capstone project, I've learned the basic HTML tags, and how they can be implemented. I implemented my HTML knowledge in this capstone project, by creating a simple resume. A resume is a skill set which describes how proficient a person on a specific mentioned skills and knowledge. A basic resume consists of the person's name, his/her personal details like photocard, address, parent's details etc. and also consist of his/her proficient skills. Here are the source codes of my capstone project-1, **Online Resume**

resume.html

```
<!DOCTYPE html>
<html>
  <head>
    <title> Venky's Resume</title>
  </head>
  <body>
    <table border="1">
      <tr>
        <td><img width="200" src="D:\venky.html\my4.jpg" alt="venky's photo"></td>
        <td>
          <h1>PANDI VENKATESWARLU</h1>
    <h2>Career Objective</h2>
    <P>I'm a proficient student pursuing a bachelor's in computer science and engineering.</P>
```

<P> “To learn and gain new experiences while utilizing my interpersonal skills to help achieve business goals.</P>

</td>

</tr>

</table>

<h2>Education</h2>

<table border="1">

<tr>

<th>COURSE</th>

<th>INSTITUTION</th>

<th>RESULT</th>

<th>YEAR</th>

</tr>

<tr>

<th>Senior Secondary (XII) </th>

<th>Sri Saraswati Junior College,Ongole</th>

<th>95.3%</th>

<th>2022</th>

</tr>

<tr>

<th>Secondary (X)</th>

<th>Cumbum Public School,Cumbum</th>

<th>100%</th>

<th>2020</th>

</tr>

</table>

<h2>Projects and Certifications</h2>

Course Completion on ‘C’ Programming language ,Jan 2023-March 2023

Successfully completed a nearly 3 months course,
which is from the 02-01-2023 to 15-03-2023 at CS CODENZ Educational Society,Gudivada,AP with 88/100%

 c language

Learn C++,Java

Aug 2023 - Sep 2023

CodeChef,w3schools.

Programming in Java course completed and achieved certificate.

Java certificate

<h3> 1.Carrer Guidance And Employement management System</h3>

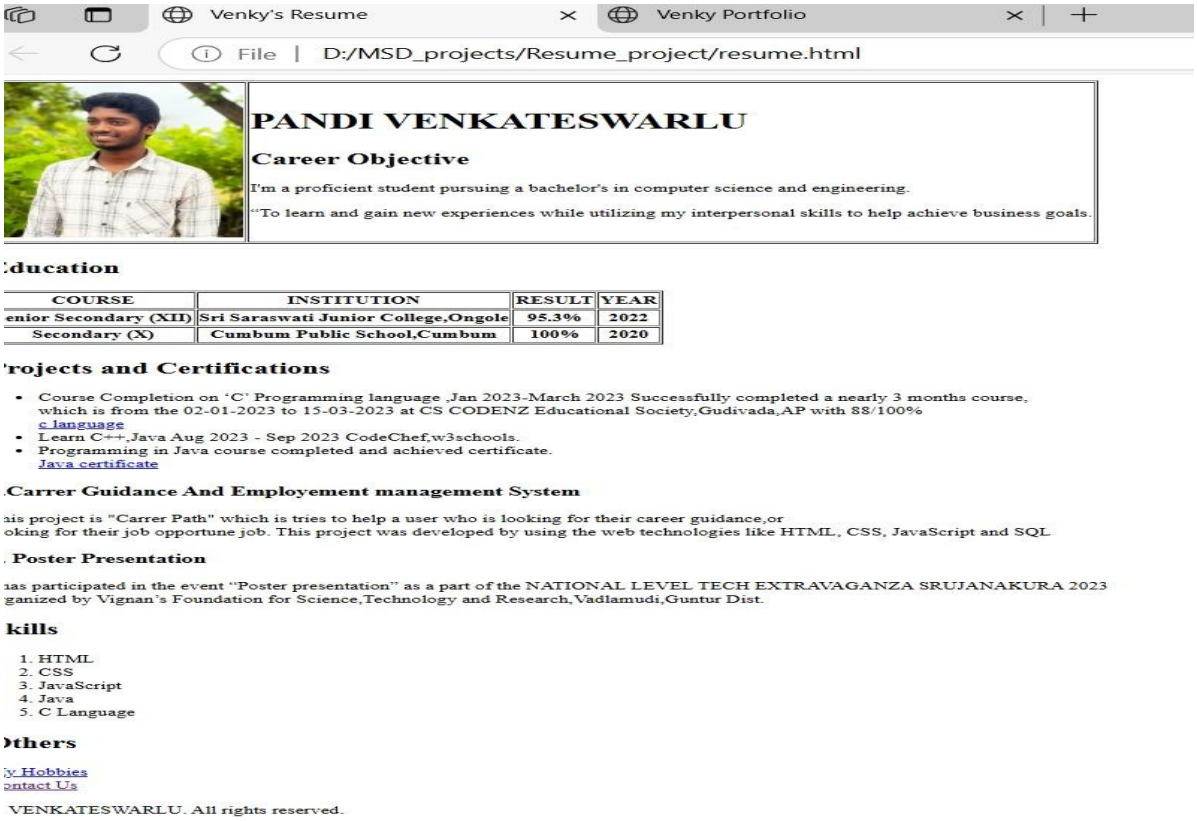
This project is "Carrer Path" which is tries to help a user who is looking for their career guidance,or
looking for their job opportune job. This project was developed by using the web technologies like HTML, CSS, JavaScript and SQL

<h3>2. Poster Presentation</h3>

I has participated in the event “Poster presentation” as a part of the NATIONAL LEVEL TECH EXTRAVAGANZA SRUJANAKURA 2023
organized by Vignan’s Foundation for Science,Technology and Research,Vadlamudi,Guntur Dist.

```
        <h2>Skills</h2>
        <ol>
            <li>HTML</li>
            <li>CSS</li>
            <li>JavaScript</li>
            <li>Java</li>
            <li>C Language</li>
        </ol>

        <h2>Others</h2>
        <a href="D:\venky.html\hobbies.html" >My Hobbies</a><br>
        <a href="D:\venky.html\contact us.html">Contact Us</a>


        <footer>
            <p>© VENKATESWARLU. All rights reserved.</p>
        </footer>

    </body>
</html>
```

Output:


Venky's Resume
Venky Portfolio
File | D:/MSD_projects/Resume_project/resume.html
PANDI VENKATESWARLU
Career Objective
I'm a proficient student pursuing a bachelor's in computer science and engineering.
"To learn and gain new experiences while utilizing my interpersonal skills to help achieve business goals.
ducation
COURSE | INSTITUTION | RESULT | YEAR
enior Secondary (XII) | Sri Saraswati Junior College,Ongole | 95.3% | 2022
Secondary (X) | Cumbum Public School,Cumbum | 100% | 2020
rojects and Certifications
Course Completion on 'C' Programming language ,Jan 2023-March 2023 Successfully completed a nearly 3 months course, which is from the 02-01-2023 to 15-03-2023 at CS CODENZ Educational Society,Gudivada,AP with 88/100%
c language
Learn C++,Java Aug 2023 - Sep 2023 CodeChef,w3schools.
Programming in Java course completed and achieved certificate.
Java certificate
Carrer Guidance And Employement management System
ais project is "Carrer Path" which is tries to help a user who is looking for their career guidance,or
oking for their job opportune job. This project was developed by using the web technologies like HTML, CSS, JavaScript and SQL
Poster Presentation
as participated in the event "Poster presentation" as a part of the NATIONAL LEVEL TECH EXTRAVAGANZA SRUJANAKURA 2023
ganized by Vignan's Foundation for Science,Technology and Research,Vadlamudi,Guntur Dist.
kills
1. HTML
2. CSS
3. JavaScript
4. Java
5. C Language
thers
y Hobbies
ontact Us
VENKATESWARLU. All rights reserved.

CAPSTONE PROJECT-2

PERSONAL WEBSITE

Objective Of Activity Done: Applying CSS to a HTML file

Detailed Report:

Before implementation of this capstone project, I've learned about Cascading Style Sheet (CSS), which is used to style the contents of the webpages. I've learnt that styling for the webpages is can be applied in three ways. i.e., Inline CSS, Internal CSS, External CSS. In this, I've considered the sample resume, which is used in capstone project-1 and applied the External CSS and created the personal webpage. This personal webpage can be uploaded in my resume, so the hiring recruiters can able to see my profile. Here are the source codes of my capstone project-2, **Personal Website resume.html**

```
<!DOCTYPE html>

<html lang="en">

<head>

   <style>

      p{          text-

align: center;

color: black;

      }


   </style>

   <meta charset="UTF-8">

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

   <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/fontawesome/6.7.1/css/all.min.css">

   <title>Venky Portfolio</title>

   <link rel="stylesheet" href="D:\venky.html\sty.css">
```

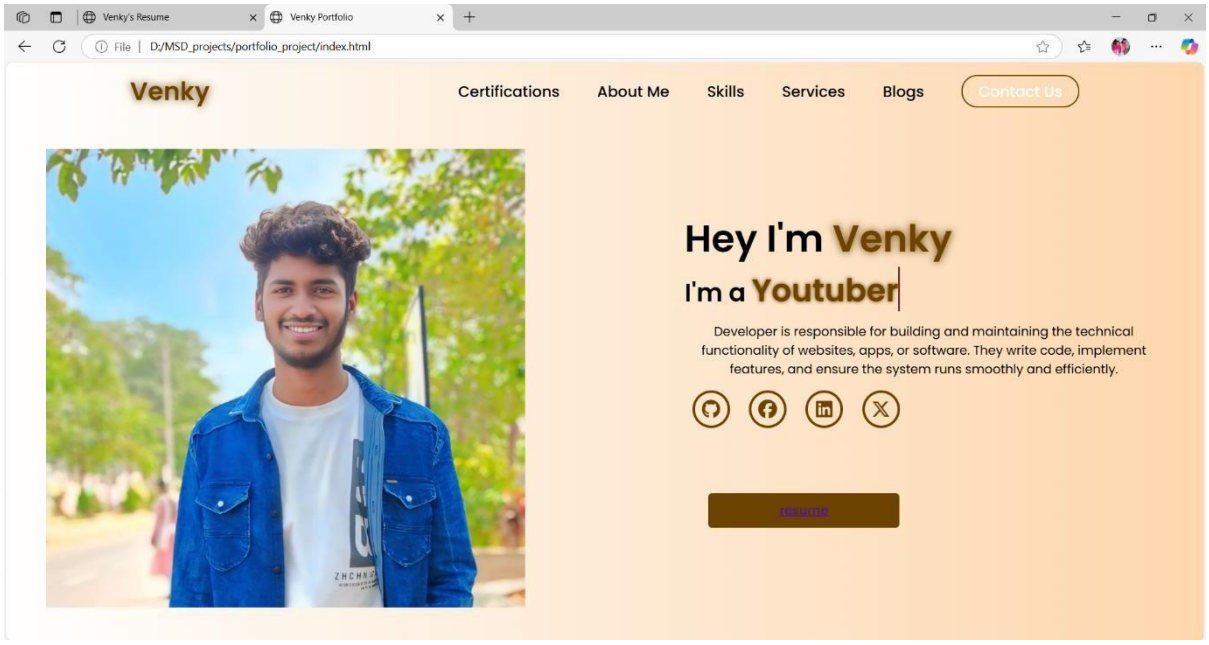
```
    <link rel="stylesheet" href="https://unpkg.com/aos@next/dist/aos.css" />
</head>
<body>
    <nav>
        <div class="nav-container">
            <div class="logo" data-aos="zoom-in" data-aos-duration="1000">
                <span>Venky</span>
            </div>
            <div class="links">
                <div class="link" data-aos="fade-up" data-aos-duration="1000" data-
aosdelay="100"><a href="D:\venky.html\certificate.html">Certifications</a></div>
                <div class="link" data-aos="fade-up" data-aos-duration="1000" data-
aosdelay="200"><a href="D:\venky.html\aboutme.html">About Me</a></div>
                <div class="link" data-aos="fade-up" data-aos-duration="1000" data-
aosdelay="300"><a href="D:\venky.html\skills.html">Skills</a></div>
                <div class="link" data-aos="fade-up" data-aos-duration="1000" data-
aosdelay="400"><a href="#">Services</a></div>
                <div class="link" data-aos="fade-up" data-aos-duration="1000" data-
aosdelay="500"><a href="#">Blogs</a></div>
                <div class="link contact-btn" data-aos="fade-up" data-aos-duration="1000"
dataaos-delay="600"><a href="D:\venky.html\contact us.html">Contact Us</a></div>
            </div>
            <i class="fa-solid fa-bars hamburg" onclick="hamburg()"></i>
        </div>
        <div class="dropdown">
            <div class="links">
                <a href="">Certifications</a>
                <a href="">About Me</a>
```

```
            <a href="D:\venky.html\skills.html">Skills</a>
            <a href="">Services</a>
            <a href="">Blogs</a>
            <a href="D:\venky.html\contact us.html">Contact Us</a>
            <i class="fa-solid fa-xmark cancel" onclick="cancel()"></i>
          </div>
        </div>
      </nav>
        <section>
          <div class="main-container">
            <div class="image" data-aos="zoom-in-middle" data-aos-duration="2500">
              <img src="D:\venky.html\venky.jpg" alt="">
            </div>
            <div class="content">
              <h1 data-aos="fade-left" data-aos-duration="1000" data-aos-delay="800">Hey I'm <span>Venky</span></h1>
              <div class="typewriter" data-aos="fade-right" data-aos-duration="1000" dataaos-delay="900">I'm a <span></span></div>
              <p data-aos="flip-up" data-aos-duration="1000" data-aosdelay="1000">Developer is responsible for building and maintaining the technical functionality of websites, apps, or software. They write code, implement features, and ensure the system runs smoothly and efficiently.</p>
              <div class="social-links" data-aos="flip-down" data-aos-duration="1000" dataaos-delay="1200">
                <a href="https://github.com/P789577" ><i class="fa-brands fagithub"></i></a>
                <a href="https://www.facebook.com/share/1B9S3V47A5/"><i class="fabrands fa-facebook"></i></a>
                <a href="https://www.linkedin.com/in/venkatesh-venky-
```

```
b98559332?utm_source=share&utm_campaign=share_via&utm_content=profile&utm_me
dium=android_app"><i class="fa-brands fa-linkedin"></i></a>

                    <a
href="https://x.com/venkateshV49269?t=yKoZeorK2iiheghGpEelKA&s=09"><i
class="fabrands fa-x-twitter"></i></a>

                </div>

                <br><br>


                 <button ><a href="D:\venky.html\Resume.html">resume</a></button>




              </div>

            </div>

        </section>

        <script src="https://unpkg.com/aos@next/dist/aos.js"></script>

        <script>

          AOS.init({offset:0});

        </script>         <script>             function hamburg(){
const navbar = document.querySelector(".dropdown")

navbar.style.transform = "translateY(0px)"

            }

            function cancel(){                 const navbar =
document.querySelector(".dropdown")

navbar.style.transform = "translateY(-500px)"

            }

        </script>
```

```
</body>

<footer>

    <p>© VENKATESWARLU. All rights reserved.</p>

</footer>

</html>
```

~~Output:~~



CAPSTONE PROJECT-3

BLOG APPLICATION

Objective Of Activity Done: Introduction to Node JS, and implementation of Node JS

Detailed Report:

Before implementation of this capstone project, I've learned about Node JS. Node JS is a frontend framework of Javascript, which is used to build the server-side applications effectively. To setup the Node JS environment in our system, we need to install it from its official website www.nodejs.org. In this Node JS, **npm** keyword is package installer, which is used to install the packages like Express JS, Mongoose, Body parser, etc. is The package Express JS is should be installed over the Node JS. In this blog application, We can create a blog, and post it to the website. These blogs are temporarily stored in an array. We can create a new blog/post, edit its content and delete it permanently. Here are the source codes of my capstone project-3, **Blog Application App.js** const express = require('express'); const app = express(); const PORT = 3030;

// Simulating a simple in-memory database let

posts = [];

app.set('view engine', 'ejs'); app.use(express.static('public'));

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

// Routes

```
// Home page - Display all posts app.get('/',
(req, res) => {     res.render('index', { posts
});
});

// Create a new post
app.get('/create', (req, res) => {
res.render('create');
}); app.post('/create', (req, res) => {
const { title, content } = req.body;
posts.push({ title, content });
res.redirect('/');
});

// Edit an existing post app.get('/edit/:id', (req,
res) => {     const post = posts[req.params.id];
res.render('edit', { post, id: req.params.id });
}); app.post('/edit/:id', (req, res) => {
const { title, content } = req.body;
posts[req.params.id] = { title, content };
res.redirect('/');
});

// Delete a post app.get('/delete/:id',
(req, res) => {
```

```
posts.splice(req.params.id, 1);
res.redirect('/');
});

app.listen(PORT, () => {     console.log(`Server running on
http://localhost:${PORT}`); });
```

index.ejs

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog Home</title>
    <link rel="stylesheet" href="/styles.css">
</head>
<body>
    <h1>Welcome to the Blog</h1>
    <a href="/create">Create New Post</a>
    <ul>
        <% posts.forEach((post, index) => { %>
            <li>
                <h2><%= post.title %></h2>
                <p><%= post.content %></p>
```

```
            <a href="/edit/<%= index %>">Edit</a>
            <a href="/delete/<%= index %>">Delete</a>
        </li>
    <% }); %>
  </ul>
</body>
</html>
```

create.ejs

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Create New Post</title>
  <link rel="stylesheet" href="/styles.css">
</head>
<body>
  <h1>Create a New Post</h1>
  <form action="/create" method="POST">
    <input type="text" name="title" placeholder="Post Title" required />
    <textarea name="content" placeholder="Post Content" required></textarea>
    <button type="submit">Create Post</button>
  </form>
  <a href="/">Go Back to Home</a>
</body>
```

```
</html>
```

new.ejs

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <title>Edit Post</title>

    <link rel="stylesheet" href="/styles.css">

</head>

<body>

    <h1>Edit Post</h1>

    <form action="/edit/<%= id %>" method="POST">

        <input type="text" name="title" value="<%= post.title %>" required />

        <textarea name="content" required><%= post.content %></textarea>

        <button type="submit">Save Changes</button>

    </form>

    <a href="/">Go Back to Home</a>

</body>

</html>
```

Styles.css body { font-

family: Arial, sans-serif;

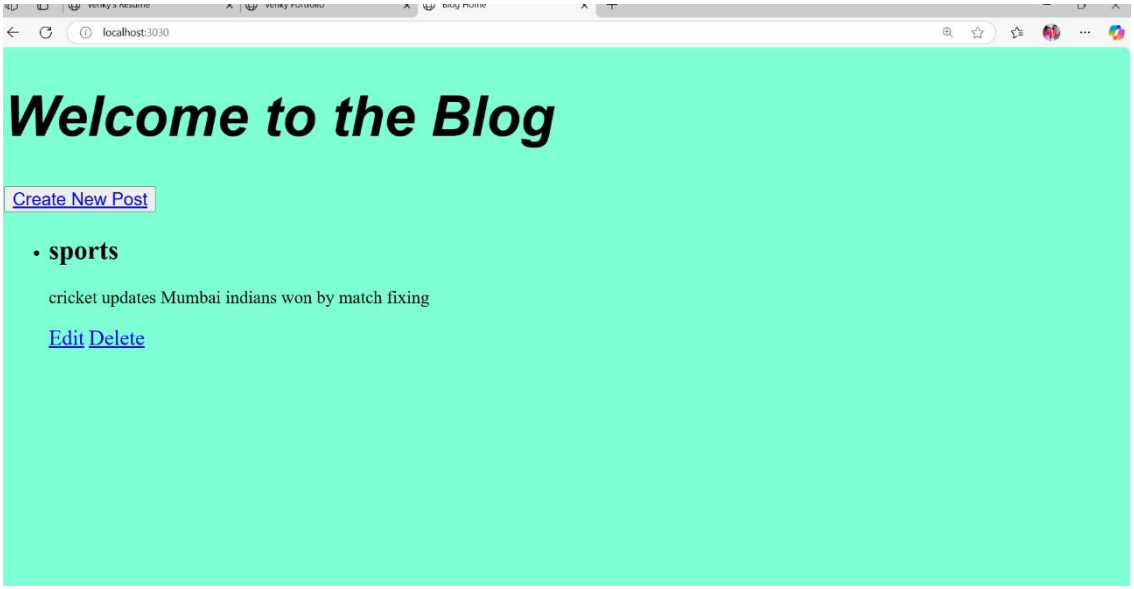
```
margin: 20px;     background-
color: #f9f9f9;
}

h1 {     color:
#333;
}

a {     text-decoration:
none;    color: #3498db;

}

ul {     list-style-type:
none;     padding: 0;
}
 li
{
   background-color: white;     margin:
10px 0;     padding: 15px;     border-radius:
5px;     box-shadow: 0 2px 4px rgba(0, 0, 0,
0.1);
}

form {
   display: flex;     flex-
direction: column;
```

```
}

input, textarea {     margin-
bottom: 10px;     padding:
10px;     font-size: 1rem;
border-radius: 5px;
border: 1px solid #ddd;
}
button {     padding: 10px;
background-color: #3498db;
color: white;     border: none;
border-radius: 5px;     cursor:
pointer;
}

button:hover {     background-
color: #2980b9;
}
```

Output:


localhost:3030
Welcome to the Blog
Create New Post


Venky's Resume
Venky Portfolio
Create New Post
localhost:3030/create
Create a New Post
sports
Mumbai indians won by match fixing
Create Post
Go Back to Home


localhost:3030
Welcome to the Blog
Create New Post
sports
cricket updates Mumbai indians won by match fixing
Edit Delete

CAPSTONE PROJECT-4

PUBLIC API

Objective Of Activity Done: Introduction to Application Programmable Interface (API), and implementation of a public API

Detailed Report:

Before implementation of this capstone project, I've learned about Application Programmable Interface (API). An API is an interface, which is used to build the communication between the end user and software program. By this API, The complex tasks can be implemented in an easy way. We can access some public APIs for free. In this capstone project, I used a public API, which can generate a random joke. Here are the source codes of my capstone project-4, **Public API**

app.js

```
const express =
require("express"); const axios =
require("axios"); const path =
require("path"); const app =
express(); app.set("view engine",
"ejs");  app.set("views",
path.join(__dirname, "views"));
app.use(express.static(path.join(__
dirname, "public")));
app.use(express.urlencoded({
extended: true })); app.get("/",
(req, res) => {    res.render("index", { joke:
null, error: null });
}); app.post("/get-joke", async (req, res)
=> {   const name = req.body.name;
```

```
  try {      const response = await
axios.get("https://v2.jokeapi.dev/joke/Any");
let joke;      if (response.data.type ===
"single") {
joke = response.data.joke;
   } else {       joke = `${response.data.setup} -
${response.data.delivery}`;
    }
    res.render("index", { joke: `Hey ${name}, here's your joke: ${joke}`, error: null });
  } catch (error) {     res.render("index", { joke: null, error: "Failed to fetch
joke. Try again!" });   }
});
app.listen(3400, () => {
console.log("Server running at port 3400");
});
```

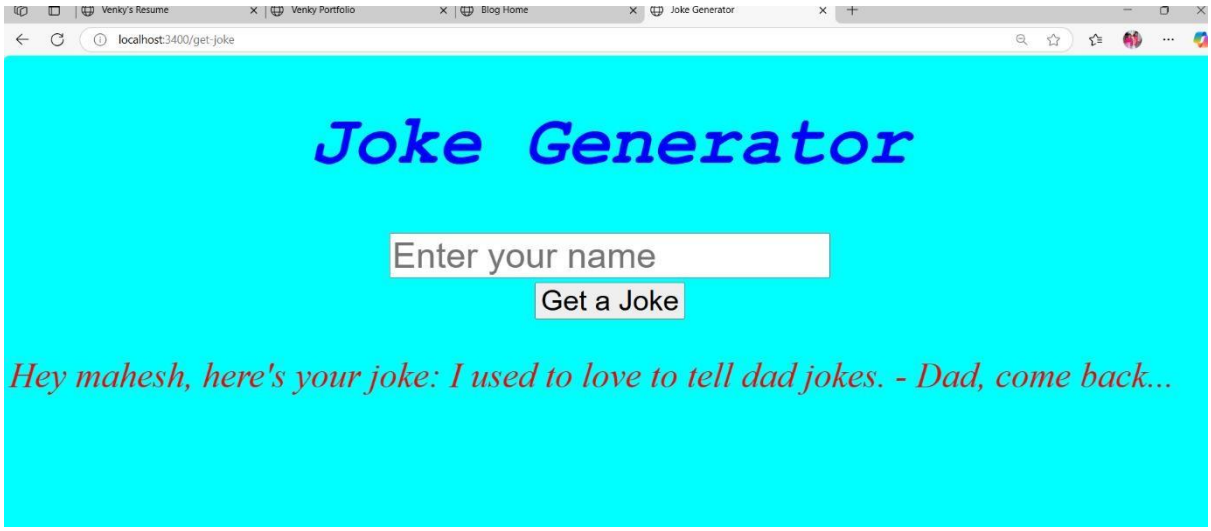
index2.ejs

```
const express =
require('express'); const app =
express(); const PORT = 3030;


let posts = [];


app.set('view engine', 'ejs');
app.use(express.static('public'));
app.use(express.urlencoded({ extended: true }));


// Routes
```

```
// Home page - Display all posts app.get('/',
(req, res) => {     res.render('index', { posts
});
});

// Create a new post
app.get('/create', (req, res) => {
res.render('create');
}); app.post('/create', (req, res) => {
const { title, content } = req.body;
posts.push({ title, content });
res.redirect('/');
});

// Edit an existing post app.get('/edit/:id', (req,
res) => {     const post = posts[req.params.id];
res.render('edit', { post, id: req.params.id });
});
app.post('/edit/:id', (req, res) => {     const
{ title, content } = req.body;
posts[req.params.id] = { title, content };
res.redirect('/');
});

// Delete a post
```

```
app.get('/delete/:id', (req, res) => {
posts.splice(req.params.id, 1);
res.redirect('/');
});

app.listen(PORT, () => {
    console.log(`Server running on http://localhost:${PORT}`); });
```

Output:


Venky's Resume
Venky Portfolio
Blog Home
Joke Generator
localhost:3400/get-joke
Joke Generator
Enter your name
Get a Joke
Hey mahesh, here's your joke: I used to love to tell dad jokes. - Dad, come back...

CAPSTONE PROJECT-5

BOOK NOTES

Objective Of Activity Done: Introduction to Postgre SQL and implementation of Book Notes application

Detailed Report:

Before implementation of this capstone project, I've learned about Postgre SQL. Postgre SQL is a relational database management system (RDBMS), where the data can be stored in table format. It is free open source, and features the transactions, ACID properties etc. Here, I implemented a book notes application, where the books that we've read that can be stored in the database of postgre SQL. We can add the new book that we read recently, and modify the details of book that we read, and delete the book details. Here are the source codes of my capstone project-5, **Book Notes index.js** const exp = require("express"); const db = require("mongoose"); const mo = require("method-override"); const path

= require("path");

const app = exp();

const Book = require("./models/book");

db.connect("mongodb://localhost:27017/bookNotesApp", {

useNewUrlParser: true, useUnifiedTopology: true,

});

app.set("view engine", "ejs"); app.set("views",

path.join(__dirname, "views"));

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

app.use(mo("_method")); app.use(exp.static("public"));

// Routes app.get("/", async (req, res) => { const { sort } =

req.query; let books = await Book.find({}); if (sort ===

```
"rating") books.sort((a, b) => b.rating - a.rating);
res.render("index", { books });
});

app.get("/books/new", (req, res) => {
res.render("new");
});

app.post("/books", async (req, res) => {    const { title, author, rating,
review, dateRead, coverId } = req.body;   await Book.create({ title,
author, rating, review, dateRead, coverId });   res.redirect("/");
});

app.get("/books/:id/edit", async (req, res) => {
const { id } = req.params;   const book = await
Book.findById(id);   res.render("edit", { book
});
});

app.put("/books/:id", async (req, res) => {
  const { id } = req.params;  const { title, author, rating, review, dateRead,
  coverId } = req.body; await
  Book.findByIdAndUpdate(id, { title, author, rating, review, dateRead, coverId });
  res.redirect("/");

});
```

```
app.delete("/books/:id", async (req, res) => {
const { id } = req.params;   await
Book.findByIdAndDelete(id);
res.redirect("/");
});

app.listen(3600, () => {    console.log("Server
running on port number 3600");
});
```

index.ejs

```
<!DOCTYPE html>
<html>
<head>
  <title>My Book Notes</title>
  <style>    body {        font-family: Arial;        margin: 2rem;        background:
linear-gradient(to right, rgba(255,0,0,0.6),rgba(255,255,255,0.4));
      }
   .book {          border: 1px solid
#ccc;            border-radius: 12px;
padding: 1rem;      margin-bottom:
1rem;      max-width: 400px;
   }  img {
width:
100px;
    }
   .A, .B{        display:inlineblock
   }
```

```
    #edit, #del{
display: inline-block;
    }
  </style>
</head>
<body>
  <h1>My Book Notes</h1>
  <div class="A">
        <a href="/books/new">Add Book</a> |
  </div>
  <div class="B">
    <a href="/?sort=rating">Sort by Rating</a>
  </div><br><br><br>
  <div class="book-list">
    <% books.forEach(book => { %>
      <div class="book">
        <h2><%= book.title %></h2>
        <p><strong>Author:</strong> <%= book.author %></p>
        <p><strong>Rating:</strong> <%= book.rating %>/10</p>
        <p><strong>Review:</strong> <%= book.review %></p>
        <p><strong>Date Read:</strong>
          <% if (book.dateRead) { %>
            <%= book.dateRead.toDateString() %>
          <% } else { %>
            Not provided
          <% } %>
```

```
        </p>

        <a id="edit" href="/books/<%= book._id %>/edit">Edit</a>  

        <form id="del" action="/books/<%= book._id %>?_method=DELETE"
method="POST">

            <button>Delete</button>

        </form>

      </div>

    <% }) %>

  </div>

</body>

</html>
```

new.ejs

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>New Book</title>

</head>

<body style="background: linear-gradient(to right,
rgba(255,0,0,0.6),rgba(255,255,255,0.4));">

    <h1>Add a New Book</h1>

    <form id="add" action="/books" method="POST">

        Title: <input id="title" name="title" /><br><br>

        Author: <input id="name" name="author" /><br><br>

        Rating: <input id="rat" name="rating" type="number" min="1" max="10"/><br><br>

        Review: <textarea id="review" name="review"></textarea><br><br>

        Date Read: <input id="date" name="dateRead" type="date" /><br><br>
```

```
      OpenLibrary Cover ID (e.g., OL12345M): <input id="coverid" name="coverId"
/><br><br><br>

      <button>Save Book</button>  <br><br>

   </form>

   <a href="/">Back</a>

</body>

</html>
```

edit.ejs

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Edit the book details</title>

</head>

<body style="background: linear-gradient(to right,
rgba(255,0,0,0.6),rgba(255,255,255,0.4));">

  <h1>Edit Book</h1>

  <form action="/books/<%= book._id %>?_method=PUT" method="POST">

    Title: <input name="title" value="<%= book.title %>" /><br><br>

    Author: <input name="author" value="<%= book.author %>" /><br><br>

    Rating: <input name="rating" type="number" min="1" max="10" value="<%= book.rating
%>" /><br><br>

    Review: <textarea name="review"><%= book.review %></textarea><br><br>

    Date Read: <input name="dateRead" type="date" value="<% if (book.dateRead) { %>

      <%= book.dateRead.toDateString() %>

    <% } else { %>

      Not provided
```

<% } %>" />

Cover ID: <input name="coverId" value="<%= book.coverId %>" />

<button>Update Book</button>

</form>

Back

</body>

</html> **./routes/book.js** const mongoose

= require("mongoose"); const

bookSchema = new mongoose.Schema({

title: String, author: String, rating: Number,

review: String, dateRead: Date, coverId: String,

});

module.exports = mongoose.model("Book", bookSchema);

Output:

Add a New Book

Title: Ramayan

Author: Sri Usha sri

Rating: 10

Review: A Good Historical book

Date Read: 08-01-2025

OpenLibrary Cover ID (e.g., OL12345M): HS01123

Save Book

Back

My Book Notes

Add Book | Sort by Rating

Ramayan

Author: Sri Usha sri

Rating: 10/10

Review: A Good Historical book

Date Read: Wed Jan 08 2025

Edit Delete

My Book Notes

Add Book | Sort by Rating

Ramayan

Author: Sri Usha sri

Rating: 10/10

Review: A Good Historical book

Date Read: Wed Jan 08 2025

Edit Delete

The beauty and the beast

Author: William Shakesphere

Rating: 7/10

Review: A good book, which defines unconditional love

Date Read: Mon Mar 24 2025

Edit Delete