



VELS



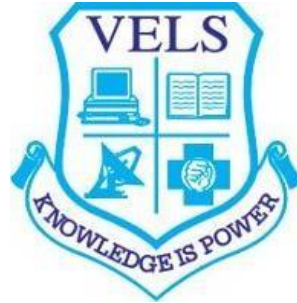
INSTITUTE OF SCIENCE, TECHNOLOGY & ADVANCED STUDIES (VISTAS)
(Deemed to be University Estd. u/s 3 of the UGC Act, 1956)

PALLAVARAM - CHENNAI

ACCREDITED BY **NAAC** WITH '**A**' GRADE

Marching Beyond 30 Years Successfully

INSTITUTION WITH **UGC 12B** STATUS



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

B.TECH - CSE - (AI & ML)

21PBAI71 - PRACTICAL - FULL STACK WEB

DEVELOPMENT LABORATORY

YEAR 2024-2025

NAME OF THE STUDENT :

REGISTER NUMBER :

COURSE :

YEAR :

SEMESTER :



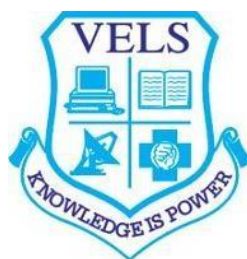
VELS



INSTITUTE OF SCIENCE, TECHNOLOGY & ADVANCED STUDIES (VISTAS)
(Deemed to be University Estd. u/s 3 of the UGC Act, 1956)

PALLAVARAM - CHENNAI

ACCREDITED BY **NAAC** WITH '**A**' GRADE
Marching Beyond 30 Years Successfully
INSTITUTION WITH **UGC 12B** STATUS



BONAFIDE CERTIFICATE

Reg. No.

--	--	--	--	--	--	--	--

This is to certify that the Bonafide Record of this Practical Work was completed by Mr./Ms..... of **B.TECH COMPUTER SCIENCE AND ENGINEERING (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)** in the Full Stack Web Development (21PBAI71) Laboratory during the academic year of 2024 – 2025.

HEAD OF THE DEPARTMENT

STAFF IN-CHARGE

Submitted for the Practical Examination held on

INTERNAL EXAMINER

EXTERNAL EXAMINER

INDEX

SNO	DATE	EXPERIMENT NAME	PAGE NO	MARKS	SIGNATURE
1		DEVELOP A PORTFOLIO WEBSITE FOR YOURSELF WHICH GIVES DETAILS ABOUT YOURSELF FOR A POTENTIAL RECRUITER.			
2		CREATE A WEB APPLICATION TO MANAGE THE TO-DO LIST OF USERS, WHERE USERS CAN LOGIN AND MANAGE THEIR TO-DO ITEMS.			
3		CREATE A SIMPLE MICRO BLOGGING APPLICATION (LIKE TWITTER) THAT ALLOWS PEOPLE TO POST THEIR CONTENT WHICH CAN BE VIEWED BY PEOPLE WHO FOLLOW THEM.			
4		CREATE A FOOD DELIVERY WEBSITE WHERE USERS CAN ORDER FOOD FROM A PARTICULAR RESTAURANT LISTED IN THE WEBSITE.			
5		TO DEVELOP AN ANDROID APPLICATION THAT MAKES USE OF DATABASE			
6		DEVELOP A LEAVE MANAGEMENT SYSTEM FOR AN ORGANIZATION WHERE USERS CAN APPLY DIFFERENT TYPES OF LEAVES SUCH AS CASUAL LEAVE AND MEDICAL LEAVE. THEY ALSO CAN VIEW THE AVAILABLE NUMBER OF DAYS.			
7		DEVELOP A SIMPLE DASHBOARD FOR PROJECT MANAGEMENT WHERE THE STATUSES OF VARIOUS TASKS ARE AVAILABLE. NEW TASKS CAN BE ADDED AND THE STATUS OF EXISTING TASKS CAN BE CHANGED AMONG PENDING. IN PROGRESS OR COMPLETED.			

SNO	DATE	EXPERIMENTAL NAME	PAGE NO	MARKS	SIGANTURE
8		DEVELOP AN ONLINE SURVEY APPLICATION WHERE A COLLECTION OF QUESTIONS IS AVAILABLE AND USERS ARE ASKED TO ANSWER ANY RANDOM 5 QUESTIONS			
9		CREATE YOUR OWN WEB APPLICATION			

Ex. No:1	DEVELOP A PORTFOLIO WEBSITE FOR YOURSELF WHICH GIVES DETAILS ABOUT YOURSELF FOR A POTENTIAL RECRUITER.

AIM:

ALGORITHM:

PROGRAM:

HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>My Portfolio</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <!-- Navbar -->
  <nav>
    <div class="logo">
      <h2>Your Name</h2>
    </div>
    <ul class="nav-links">
      <li><a href="#home">Home</a></li>
      <li><a href="#about">About</a></li>
      <li><a href="#projects">Projects</a></li>
      <li><a href="#contact">Contact</a></li>
    </ul>
  </nav>

  <!-- Home Section -->
  <section id="home" class="home-section">
    <div class="intro">
      <h1>Hello, I'm <span>Your Name</span></h1>
      <p>Full-Stack Developer | AI Enthusiast</p>
      <a href="#projects" class="btn">Explore My Work</a>
    </div>
  </section>

  <!-- About Section -->
  <section id="about" class="about-section">
```

About Me

I'm a passionate full-stack developer with a deep interest in AI.

I specialize in building responsive web applications and integrating AI/ML solutions into real-world applications.

<section id="projects" class="projects-section">

Projects

<div class="projects-container">

<div class="project-card">

SMART TRAFFIC VIOLATION DETECTION

<p>A system for detecting helmet violations and generating e-challans using YOLO and OCR.</p>

</div>

<div class="project-card">

NEURO-VISION TUMOR TRACKER

<p>An AI-powered solution for brain tumor detection using MRI scans and deep learning models.</p>

</div>

</div>

</section>

<section id="contact" class="contact-section">

Contact Me

<p>Feel free to reach out to me via email: youremail@example.com</p>

>

</section>

<footer>

<p>© 2024 Your Name. All rights reserved.</p>

</footer>

</body>

</html>

CSS (styles.css):

```
/* Global Styles */
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  background-color: #f0f0f0;
  color: #333;
}
/* Navbar Styles */
nav {
  display: flex;
  justify-content: space-between;
  align-items: center;
  background-color: #333;
  color: #fff;
  padding: 1rem 2rem;
}
.nav-links {
  list-style: none;
  display: flex;
}
.nav-links li {
  margin-left: 20px;
}
.nav-links a {
  color: white;
  text-decoration: none;
  font-weight: bold;
}
.nav-links a:hover {
  color: #ff6b6b;
}
/* Home Section */
.home-section {
  height: 100vh;
  background: linear-gradient(rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5)),
    url('your-photo.jpg') center/cover no-repeat;
```



```
    display: flex;
    justify-content: center;
    align-items: center;
    text-align: center;
    color: white;
}

.home-section h1 {
    font-size: 3rem;
    margin-bottom: 1rem;
}

.home-section h1 span {
    color: #ff6b6b;
}

.home-section p {
    font-size: 1.2rem;
    margin-bottom: 1.5rem;
}

.btn {
    padding: 10px 20px;
    background-color: #ff6b6b;
    color: white;
    text-decoration: none;
    font-weight: bold;
    border-radius: 5px;
}

.btn:hover {
    background-color: #ff3d3d;
}

/* About Section */
.about-section {
    padding: 50px;
    background-color: #fff;
    text-align: center;
}
```

```
.about-section h2 {  
  font-size: 2.5rem;  
  margin-bottom: 20px;  
}
```

```
.about-section p {  
  font-size: 1.2rem;  
  width: 70%;  
  margin: 0 auto;  
}
```

```
/* Projects Section */  
.projects-section {  
  padding: 50px;  
  background-color: #f9f9f9;  
  text-align: center;  
}
```

```
.projects-section h2 {  
  font-size: 2.5rem;  
  margin-bottom: 30px;  
}
```

```
.projects-container {  
  display: flex;  
  justify-content: space-around;  
  flex-wrap: wrap;  
}
```

```
.project-card {  
  background-color: white;  
  padding: 20px;  
  margin: 10px;  
  border-radius: 10px;  
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  
  width: 300px;  
}
```

```
.project-card h3 {  
  color: #ff6b6b;
```

```
    margin-bottom: 10px;
}
```

```
.project-card p {
    font-size: 1rem;
    color: #666;
}
```

```
/* Contact Section */
.contact-section {
    padding: 50px;
    background-color: #fff;
    text-align: center;
}
```

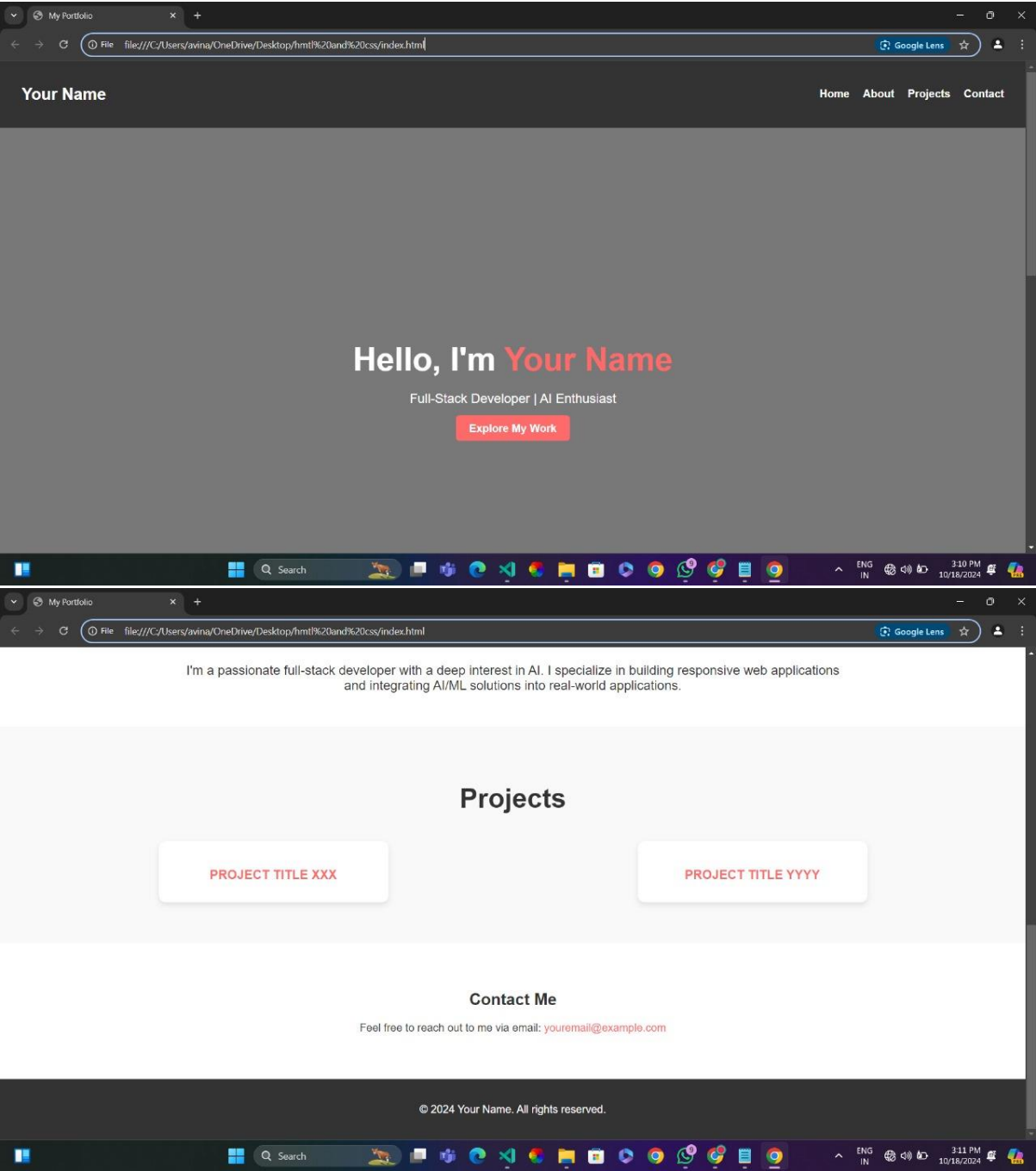
```
.contact-section h2 {
    font-size: 2.5rem;
    margin-bottom: 20px;
}
```

```
.contact-section p a {
    color: #ff6b6b;
    text-decoration: none;
}
```

```
.contact-section p a:hover {
    text-decoration: underline;
}
```

```
/* Footer */
footer {
    background-color: #333;
    color: white;
    text-align: center;
    padding: 20px;
}
```

OUTPUT:



RESULT:

Ex. No: 2	CREATE A WEB APPLICATION TO MANAGE THE TO-DO LIST OF USERS, WHERE USERS CAN LOGIN AND MANAGE THEIR TO-DO ITEMS.

AIM:

ALGORITHM:

PROGRAM:

Main To-Do Page (index.html):

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>To-Do List</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="container">
    <h1>To-Do List</h1>
    <div class="todo-container">
      <input type="text" id="new-todo" placeholder="Enter a new task">
      <button id="add-todo-btn">Add</button>
      <ul id="todo-list"></ul>
    </div>
    <button id="logout-btn">Logout</button>
  </div>
</body>
</html>
```

LOGIN PAGE (login.html):

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Login</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="login-container">
    <h1>Login</h1>
```

```
<form action="#">
  <input type="text" placeholder="Username" required>
  <input type="password" placeholder="Password" required>
  <button type="submit">Login</button>
</form>
<p>Don't have an account? <a href="register.html">Register
here</a></p>
</div>
</body>
</html>
```

REGISTER PAGE (register.html):

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Register</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="register-container">
    <h1>Register</h1>
    <form action="#">
      <input type="text" placeholder="Username" required>
      <input type="password" placeholder="Password" required>
      <button type="submit">Register</button>
    </form>
    <p>Already have an account? <a href="login.html">Login
here</a></p>
  </div>
</body>
</html>
```

CSS (styles.css):

```
/* General Styles */
body {
  font-family: Arial, sans-serif;
  background-color: #f4f4f4;
  margin: 0;
  padding: 0;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}

.container {
  text-align: center;
  background-color: #fff;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  width: 300px;
}

h1 {
  color: #333;
  margin-bottom: 20px;
}

/* To-Do Styles */
.todo-container {
  margin-bottom: 20px;
}

input[type="text"] {
  padding: 10px;
  width: calc(100% - 24px);
  margin-bottom: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
```



```
}
button {
  padding: 10px;
  width: 100%;
  background-color: #28a745;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

button:hover {
  background-color: #218838;
}

ul {
  list-style: none;
  padding: 0;
  margin: 0;
}

ul li {
  background-color: #f9f9f9;
  padding: 10px;
  margin-bottom: 5px;
  border: 1px solid #ddd;
  border-radius: 5px;
  display: flex;
  justify-content: space-between;
}

/* Login and Register Styles */
.login-container, .register-container {
  background-color: #fff;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  width: 300px;
  text-align: center;
}
```

```
}  
input[type="text"], input[type="password"] {  
  width: calc(100% - 24px);  
  padding: 10px;  
  margin-bottom: 10px;  
  border: 1px solid #ccc;  
  border-radius: 5px;  
}
```

```
form button {  
  padding: 10px;  
  width: 100%;  
  background-color: #007bff;  
  color: white;  
  border: none;  
  border-radius: 5px;  
  cursor: pointer;  
}
```

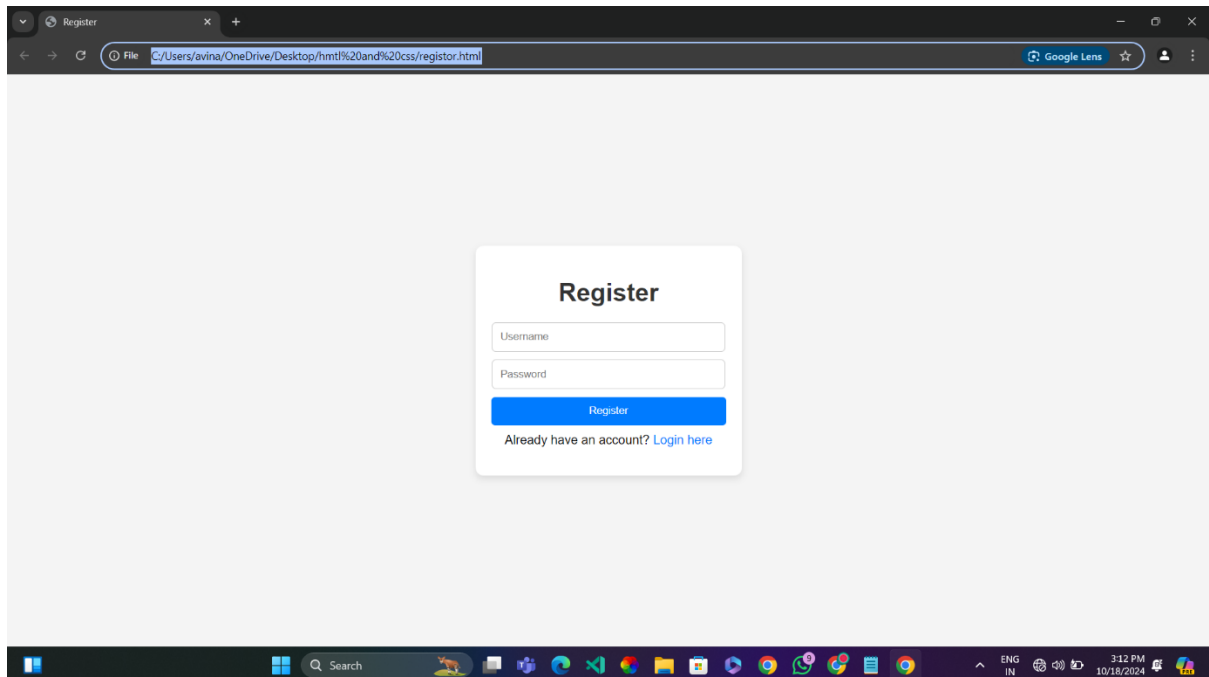
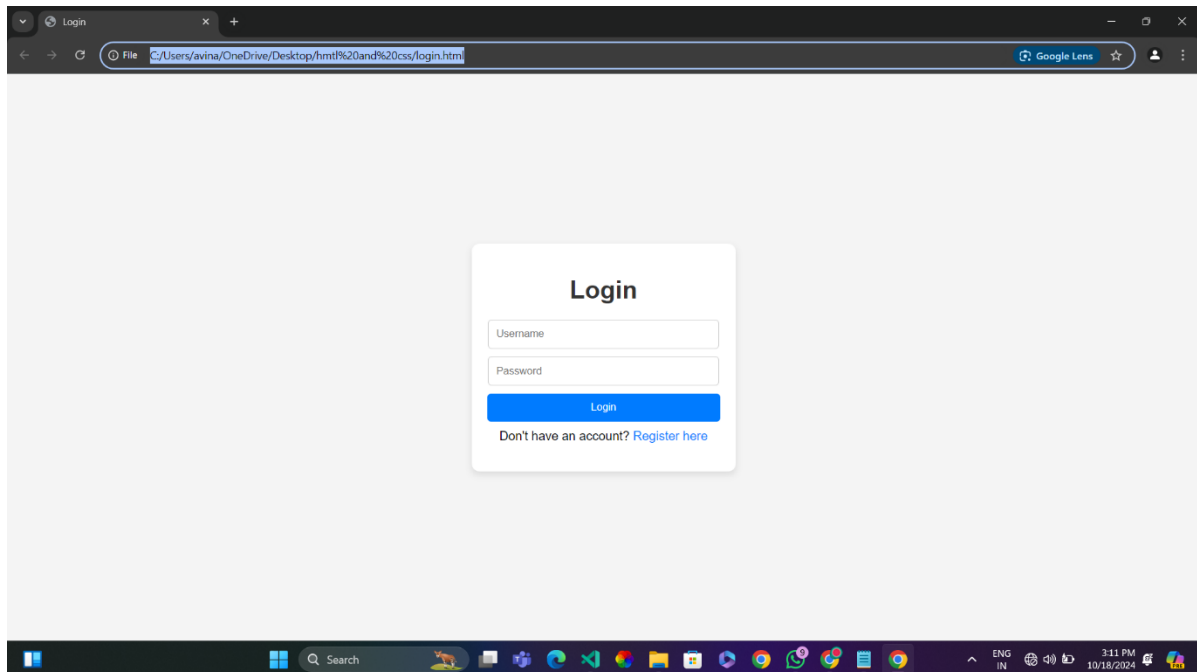
```
form button:hover {  
  background-color: #0056b3;  
}
```

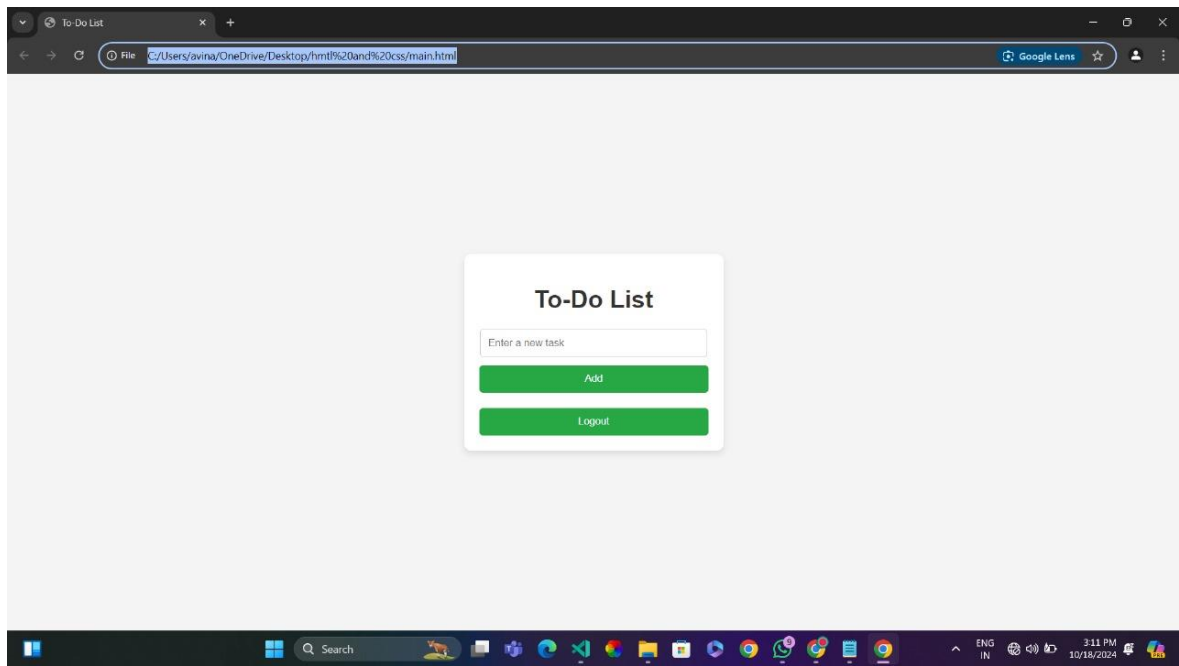
```
p {  
  margin-top: 10px;  
}
```

```
a {  
  color: #007bff;  
  text-decoration: none;  
}
```

```
a:hover {  
  text-decoration: underline;  
}
```

OUTPUT:





RESULT:

Ex. No: 3	CREATE A SIMPLE MICRO BLOGGING APPLICATION (LIKE TWITTER) THAT ALLOWS PEOPLE TO POST THEIR CONTENT WHICH CAN BE VIEWED BY PEOPLE WHO FOLLOW THEM.

AIM:

ALGORITHM:

PROGRAM:
BACKEND (Node.js & Express.js):
server.js:

```
const express = require('express');
const mongoose = require('mongoose');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const bodyParser = require('body-parser');
const app = express();
app.use(bodyParser.json());
app.use(express.static('public'));
const jwtSecret = "supersecretkey";
// Connect to MongoDB
mongoose.connect('mongodb://localhost/microblog', { useNewUrlParser: true,
useUnifiedTopology: true })
    .then(() => console.log('Connected to MongoDB...'))
    .catch(err => console.error('Could not connect to MongoDB...', err));
// User Schema
const userSchema = new mongoose.Schema({
    username: { type: String, unique: true },
    password: String,
    followers: [{ type: mongoose.Schema.Types.ObjectId, ref: 'User' }],
    following: [{ type: mongoose.Schema.Types.ObjectId, ref: 'User' }]
});
const User = mongoose.model('User', userSchema);
// Post Schema
const postSchema = new mongoose.Schema({
    content: String,
    user: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
    date: { type: Date, default: Date.now }
```

```
});  
const Post = mongoose.model('Post', postSchema);  
  
// Register user  
app.post('/register', async (req, res) => {  
  const { username, password } = req.body;  
  const hashedPassword = await bcrypt.hash(password, 10);  
  const user = new User({ username, password: hashedPassword });  
  await user.save();  
  res.json({ message: 'User registered successfully!' });  
});  
  
// Login user  
app.post('/login', async (req, res) => {  
  const { username, password } = req.body;  
  const user = await User.findOne({ username });  
  if (!user) return res.status(400).json({ message: 'Invalid username or password' });  
  const isMatch = await bcrypt.compare(password, user.password);  
  if (!isMatch) return res.status(400).json({ message: 'Invalid username or password' });  
  const token = jwt.sign({ userId: user._id }, jwtSecret);  
  res.json({ token, userId: user._id });  
});  
  
// Post content  
app.post('/post', async (req, res) => {  
  const { content, token } = req.body;  
  const decoded = jwt.verify(token, jwtSecret);  
  const post = new Post({  
    content,  
    user: decoded.userId  
  });  
  await post.save();  
  res.json({ message: 'Post created successfully!' });  
});
```

```
});  
  
// Follow a user  
app.post('/follow', async (req, res) => {  
  const { token, followUserId } = req.body;  
  const decoded = jwt.verify(token, jwtSecret);  
  const user = await User.findById(decoded.userId);  
  const followUser = await User.findById(followUserId);  
  if (!user.following.includes(followUserId)) {  
    user.following.push(followUserId);  
    followUser.followers.push(user._id);  
  }  
  await user.save();  
  await followUser.save();  
  res.json({ message: `You are now following ${followUser.username}` });  
});  
  
// Fetch posts from people the user follows  
app.get('/feed', async (req, res) => {  
  const token = req.headers.authorization.split(' ')[1];  
  const decoded = jwt.verify(token, jwtSecret);  
  const user = await User.findById(decoded.userId).populate('following');  
  const posts = await Post.find({ user: { $in: user.following } })  
    .populate('user')  
    .sort({ date: -1 });  
  res.json(posts);  
});  
  
app.listen(3000, () => console.log('Server started on port 3000'));
```


FRONTEND (HTML/CSS/JavaScript)

index.html:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Microblogging App</title>

  <link rel="stylesheet" href="style.css">

</head>

<body>

  <div class="container">

    <h1>Microblogging Application</h1>

    <div id="auth">

      <h2>Register</h2>

      <input type="text" id="regUsername" placeholder="Username">

      <input type="password" id="regPassword" placeholder="Password">

      <button id="registerBtn">Register</button>

      <h2>Login</h2>

      <input type="text" id="loginUsername" placeholder="Username">

      <input type="password" id="loginPassword" placeholder="Password">

      <button id="loginBtn">Login</button>

    </div>

    <div id="app" style="display: none;">

      <h2>Create a Post</h2>

      <input type="text" id="postContent" placeholder="What's on your mind?">

      <button id="postBtn">Post</button>

      <h2>Your Feed</h2>

      <div id="feed"></div>

    </div>

  </div>

</body>
```

```
</div>
<script src="app.js"></script>
</body>
</html>
```

CSS (styles.css):

```
body {
  font-family: Arial, sans-serif;
  background-color: #f9f9f9;
  padding: 20px;
}
.container {
  width: 60%;
  margin: auto;
  background-color: white;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
h1, h2 {
  text-align: center;
}
input {
  width: 100%;
  padding: 10px;
  margin: 5px 0;
  font-size: 16px;
  border: 1px solid #ccc;
  border-radius: 5px;
}
```

```
button {  
  width: 100%;  
  padding: 10px;  
  background-color: #28a745;  
  color: white;  
  border: none;  
  border-radius: 5px;  
  cursor: pointer;  
}  
button:hover {  
  background-color: #218838;  
}  
#feed {  
  margin-top: 20px;  
}
```

FRONTEND JAVASCRIPT (Client-Side Logic):

app.js:

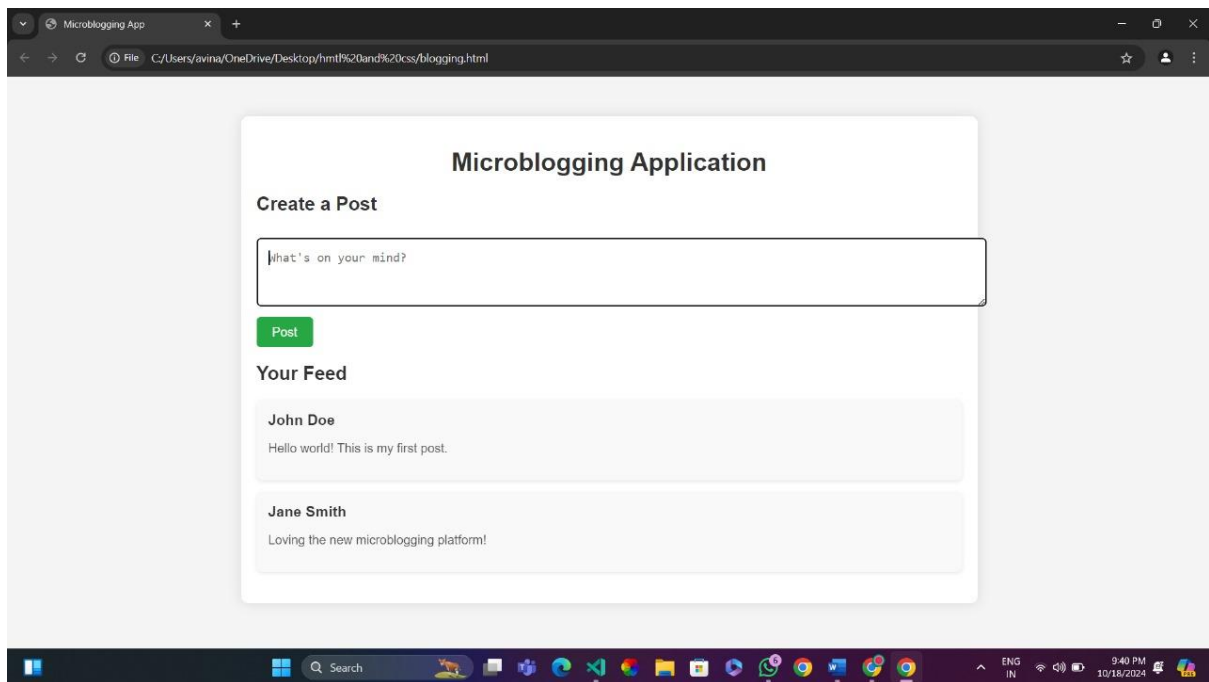
```
let token = "";  
document.getElementById('registerBtn').addEventListener('click', async () => {  
  const username = document.getElementById('regUsername').value;  
  const password = document.getElementById('regPassword').value;  
  const response = await fetch('/register', {  
    method: 'POST',  
    headers: { 'Content-Type': 'application/json' },  
    body: JSON.stringify({ username, password })  
  });  
  const result = await response.json();  
  alert(result.message);  
});
```

```
document.getElementById('loginBtn').addEventListener('click', async () => {
  const username = document.getElementById('loginUsername').value;
  const password = document.getElementById('loginPassword').value;
  const response = await fetch('/login', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ username, password })
  });
  const result = await response.json();
  token = result.token;
  document.getElementById('auth').style.display = 'none';
  document.getElementById('app').style.display = 'block';
  fetchFeed();
});

document.getElementById('postBtn').addEventListener('click', async () => {
  const content = document.getElementById('postContent').value;
  await fetch('/post', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ content, token })
  });
  document.getElementById('postContent').value = '';
  fetchFeed();
});

async function fetchFeed() {
  const response = await fetch('/feed', {
    headers: { Authorization: `Bearer ${token}` }
  });
  const posts = await response.json();
  const feedDiv = document.getElement
```

OUTPUT:



RESULT:

Ex. No: 4	CREATE A SIMPLE MICRO BLOGGING APPLICATION (LIKE TWITTER) THAT ALLOWS PEOPLE TO POST THEIR CONTENT WHICH CAN BE VIEWED BY PEOPLE WHO FOLLOW THEM.

AIM:

ALGORITHM:

PROGRAM:

BACKEND (Node.js & Express.js):

server.js:

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const app = express();
app.use(bodyParser.json());
app.use(express.static('public'));

// Connect to MongoDB
mongoose.connect('mongodb://localhost/food_delivery', { useNewUrlParser: true,
useUnifiedTopology: true })

    .then(() => console.log('Connected to MongoDB...'))
    .catch(err => console.error('Could not connect to MongoDB...', err));

// Restaurant Schema
const restaurantSchema = new mongoose.Schema({
    name: String,
    foodItems: [{ name: String, price: Number }]
});
const Restaurant = mongoose.model('Restaurant', restaurantSchema);

// Order Schema
const orderSchema = new mongoose.Schema({
    userName: String,
    restaurantName: String,
    foodItems: [{ name: String, quantity: Number, price: Number }],
    totalPrice: Number
});
const Order = mongoose.model('Order', orderSchema);

// Fetch all restaurants
app.get('/restaurants', async (req, res) => {
    const restaurants = await Restaurant.find();
    res.json(restaurants);
});
```

```
});  
// Place an order  
app.post('/order', async (req, res) => {  
  const { userName, restaurantName, foodItems } = req.body;  
  // Calculate total price  
  let totalPrice = foodItems.reduce((total, item) => total + (item.price * item.quantity), 0);  
  const order = new Order({  
    userName,  
    restaurantName,  
    foodItems,  
    totalPrice  
  });  
  await order.save();  
  res.json({ message: "Order placed successfully!", order });  
});  
app.listen(3000, () => console.log('Server started on port 3000'));
```

FRONTEND (HTML/CSS/JavaScript):

index.html:

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>Food Delivery Website</title>  
  <link rel="stylesheet" href="style.css">  
</head>  
<body>  
  <div class="container">  
    <h1>Food Delivery</h1>  
    <!-- Restaurant List -->
```



```
<div id="restaurantList"></div>

<!-- Selected Restaurant Food Items -->

<div id="foodItems" style="display: none;">
  <h2 id="restaurantName"></h2>
  <div id="menu"></div>
  <button id="viewCartBtn" onclick="viewCart()">View Cart</button>
</div>

<!-- Cart -->

<div id="cart" style="display: none;">
  <h2>Cart</h2>
  <div id="cartItems"></div>
  <input type="text" id="userName" placeholder="Enter your name">
  <button onclick="placeOrder()">Place Order</button>
</div>
</div>

<script src="app.js"></script>
</body>
</html>
```

CSS (styles.css):

```
body {
  font-family: Arial, sans-serif;
  background-color: #f4f4f4;
  padding: 20px;
}

.container {
  width: 60%;
  margin: auto;
  background-color: white;
  padding: 20px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  border-radius: 10px;
```

```
}  
h1, h2 {  
  text-align: center;  
  color: #333;  
}  
#restaurantList, #menu, #cartItems {  
  display: flex;  
  flex-direction: column;  
}  
button {  
  padding: 10px;  
  margin: 5px;  
  background-color: #28a745;  
  color: white;  
  border: none;  
  cursor: pointer;  
}  
button:hover {  
  background-color: #218838;  
}
```

FRONTEND JAVASCRIPT (Client-Side Logic):

app.js:

```
let selectedRestaurant = null;  
let cart = [];  
// Fetch and display restaurants  
async function fetchRestaurants() {  
  const response = await fetch('/restaurants');  
  const restaurants = await response.json();  
  const restaurantList = document.getElementById('restaurantList');  
  restaurantList.innerHTML = `<h2>Select a Restaurant</h2>`;
```

```

restaurants.forEach(restaurant => {

  const div = document.createElement('div');

  div.innerHTML = `

    <button onclick="selectRestaurant('${restaurant.name}')">${restaurant.name}</button>

  `;

  restaurantList.appendChild(div);

});
}

// Select a restaurant and display its food items
async function selectRestaurant(name) {

  selectedRestaurant = name;

  const response = await fetch('/restaurants');
  const restaurants = await response.json();
  const restaurant = restaurants.find(r => r.name === name);
  const foodItemsDiv = document.getElementById('foodItems');
  const menuDiv = document.getElementById('menu');
  const restaurantName = document.getElementById('restaurantName');
  foodItemsDiv.style.display = 'block';
  restaurantName.innerHTML = name;
  menuDiv.innerHTML = "";
  restaurant.foodItems.forEach(item => {

    const itemDiv = document.createElement('div');

    itemDiv.innerHTML = `

      <span>${item.name} - ${item.price}</span>

      <input type="number" min="1" value="1" id="quantity_${item.name}">

      <button onclick="addToCart('${item.name}', ${item.price})">Add to Cart</button>

    `;

    menuDiv.appendChild(itemDiv);

  });
}

function addToCart(name, price) {

  const quantity = document.getElementById(`quantity_${name}`).value;

```

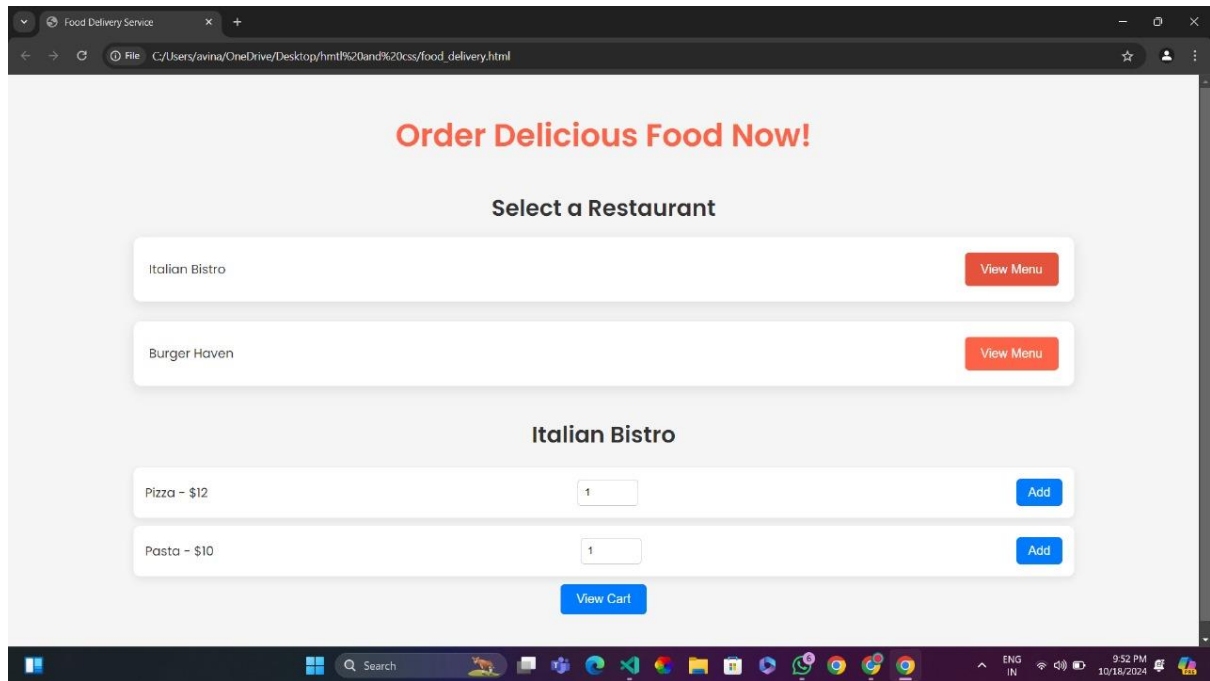
```
    cart.push({ name, price, quantity: parseInt(quantity) });
    alert(`${name} added to cart`);
}

function viewCart() {
    const cartDiv = document.getElementById('cart');
    const cartItemsDiv = document.getElementById('cartItems');
    cartDiv.style.display = 'block';
    cartItemsDiv.innerHTML = '';
    cart.forEach(item => {
        const itemDiv = document.createElement('div');
        itemDiv.innerHTML = `<span>${item.name} x${item.quantity} - ${item.price *
item.quantity}</span>`;
        cartItemsDiv.appendChild(itemDiv);
    });
}

async function placeOrder() {
    const userName = document.getElementById('userName').value;
    const response = await fetch('/order', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({
            userName,
            restaurantName: selectedRestaurant,
            foodItems: cart
        })
    });
    const result = await response.json();
    alert(result.message);
    cart = [];
    document.getElementById('cart').style.display = 'none';
}

fetchRestaurants();
```

OUTPUT:



RESULT:

Ex. No: 5	DEVELOP A CLASSIFIEDS WEB APPLICATION TO BUY AND SELL USED PRODUCTS, REST API WITH NODE.

AIM:

ALGORITHM:

PROGRAM:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Product Marketplace</title>
    <style>
      * {
        margin: 0;
        padding: 0;
        box-sizing: border-box;
        font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
      }

      body {
        background-color: #f5f5f5;
      }

      .container {
        max-width: 1200px;
        margin: 0 auto;
        padding: 20px;
      }

      header {
        background-color: #2c3e50;
        color: white;
        padding: 1rem;
        margin-bottom: 2rem;
      }

      .search-bar {
        display: flex;
        gap: 10px;
        margin-bottom: 20px;
      }

      input,
      select,
      button {
        padding: 10px;
        border: 1px solid #ddd;
        border-radius: 4px;
```

```
}
```

```
button {  
  background-color: #3498db;  
  color: white;  
  border: none;  
  cursor: pointer;  
  transition: background-color 0.3s;  
}
```

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

```
.products-grid {  
  display: grid;  
  grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));  
  gap: 20px;  
}
```

```
.product-card {  
  background-color: white;  
  border-radius: 8px;  
  padding: 15px;  
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);  
}
```

```
.product-card img {  
  width: 100%;  
  height: 200px;  
  object-fit: cover;  
  border-radius: 4px;  
  margin-bottom: 10px;  
}
```

```
.product-card h3 {  
  margin-bottom: 10px;  
  color: #2c3e50;  
}
```

```
.price {  
  font-size: 1.2rem;  
  color: #27ae60;  
  font-weight: bold;  
  margin-bottom: 10px;  
}
```



```
.modal {
  display: none;
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, 0.5);
}
.modal-content {
  background-color: white;
  padding: 20px;
  border-radius: 8px;
  width: 90%;
  max-width: 500px;
  margin: 50px auto;
}

.form-group {
  margin-bottom: 15px;
}

.form-group label {
  display: block;
  margin-bottom: 5px;
}

.form-group input,
.form-group textarea,
.form-group select {
  width: 100%;
}

.close {
  float: right;
  cursor: pointer;
  font-size: 1.5rem;
}
</style>
</head>
<body>
<header>
  <div class="container">
    <h1>Product Marketplace</h1>
  </div>
</header>
```

```
<div class="container">
  <div class="search-bar">
    <input type="text" id="searchInput" placeholder="Search products..." />
    <select id="categoryFilter">
      <option value="">All Categories</option>
      <option value="electronics">Electronics</option>
      <option value="furniture">Furniture</option>
      <option value="clothing">Clothing</option>
      <option value="books">Books</option>
    </select>
    <button onclick="searchProducts()">Search</button>
    <button onclick="showAddProductModal()">Add Product</button>
  </div>

  <div class="products-grid" id="productsGrid"></div>
</div>
<div id="addProductModal" class="modal">
  <div class="modal-content">
    <span class="close" onclick="closeModal()">&times;</span>
    <h2>Add New Product</h2>
    <form id="addProductForm">
      <div class="form-group">
        <label for="title">Title</label>
        <input type="text" id="title" required />
      </div>
      <div class="form-group">
        <label for="description">Description</label>
        <textarea id="description" required></textarea>
      </div>
      <div class="form-group">
        <label for="price">Price</label>
        <input type="number" id="price" required />
      </div>
      <div class="form-group">
        <label for="category">Category</label>
        <select id="category" required>
          <option value="electronics">Electronics</option>
          <option value="furniture">Furniture</option>
          <option value="clothing">Clothing</option>
          <option value="books">Books</option>
        </select>
      </div>
      <div class="form-group">
        <label for="condition">Condition</label>
        <select id="condition" required>
          <option value="new">New</option>
        </select>
      </div>
    </form>
  </div>
</div>
```

```

        <option value="like-new">Like New</option>
        <option value="good">Good</option>
        <option value="fair">Fair</option>
    </select>
</div>
<div class="form-group">
    <label for="contact">Contact Information</label>
    <input type="text" id="contact" required />
</div>
<div class="form-group">
    <label for="image">Image URL</label>
    <input type="url" id="image" />
</div>
<button type="submit">Add Product</button>
</form>
</div>
</div>

<script>
const API_URL = "http://localhost:3000/api";

document.addEventListener("DOMContentLoaded", loadProducts);

async function loadProducts() {
    try {
        const response = await fetch(`${API_URL}/products`);
        const products = await response.json();
        displayProducts(products);
    } catch (error) {
        console.error("Error loading products:", error);
    }
}

async function searchProducts() {
    const searchTerm = document.getElementById("searchInput").value;
    const category = document.getElementById("categoryFilter").value;

    try {
        let url = `${API_URL}/products`;
        if (searchTerm) {
            url = `${API_URL}/search?q=${searchTerm}`;
        }
        if (category) {
            url += `${searchTerm ? "&" : "?"}category=${category}`;
        }
    }
}

```

```

    const response = await fetch(url);
    const products = await response.json();
    displayProducts(products);
  } catch (error) {
    console.error("Error searching products:", error);
  }
}

function displayProducts(products) {
  const grid = document.getElementById("productsGrid");
  grid.innerHTML = "";

  products.forEach((product) => {
    const card = document.createElement("div");
    card.className = "product-card";
    card.innerHTML = `
      
      <h3>${product.title}</h3>
      <p class="price">${product.price}</p>
      <p>${product.description.substring(0, 100)}...</p>
      <p><strong>Condition:</strong> ${product.condition}</p>
      <p><strong>Contact:</strong> ${product.contact}</p>
    `;
    grid.appendChild(card);
  });
}

function showAddProductModal() {
  document.getElementById("addProductModal").style.display = "block";
}

function closeModal() {
  document.getElementById("addProductModal").style.display = "none";
}

document
  .getElementById("addProductForm")
  .addEventListener("submit", async (e) => {
    e.preventDefault();

    const productData = {
      title: document.getElementById("title").value,
      description: document.getElementById("description").value,
      price: Number(document.getElementById("price").value),
    };
  });

```

```

        category: document.getElementById("category").value,
        condition: document.getElementById("condition").value,
        contact: document.getElementById("contact").value,
        image: document.getElementById("image").value,
    };

    try {
        const response = await fetch(`${API_URL}/products`, {
            method: "POST",
            headers: {
                "Content-Type": "application/json",
            },
            body: JSON.stringify(productData),
        });

        if (response.ok) {
            closeModal();
            loadProducts();
            e.target.reset();
        }
    } catch (error) {
        console.error("Error adding product:", error);
    }
});

window.onclick = function (event) {
    const modal = document.getElementById("addProductModal");
    if (event.target === modal) {
        closeModal();
    }
};
</script>
</body>
</html>

```

IN TERMINAL RUN THIS COMMAND

- npm init

- npm i express mongoose cors

THEN CREATE index.js FILE IN FOLDER

```

-----NODE JS BACKEND CODE "index.js"-----
const express = require("express");
const mongoose = require("mongoose");
const cors = require("cors");

```

```
const app = express();
app.use(express.json());
app.use(cors());
mongoose.connect("mongodb://localhost:27017/marketplace", {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});
const ProductSchema = new mongoose.Schema({
  title: { type: String, required: true },
  description: { type: String, required: true },
  price: { type: Number, required: true },
  category: { type: String, required: true },
  contact: { type: String, required: true },
  image: String,
  createdAt: { type: Date, default: Date.now },
});

const Product = mongoose.model("Product", ProductSchema);

app.post("/api/products", async (req, res) => {
  try {
    const product = new Product(req.body);
    await product.save();
    res.status(201).send(product);
  } catch (error) {
    res.status(400).send({ error: error.message });
  }
});

app.get("/api/products", async (req, res) => {
  try {
    const filters = {};

    if (req.query.category) filters.category = req.query.category;
    if (req.query.minPrice)
      filters.price = { $gte: parseFloat(req.query.minPrice) };
    if (req.query.maxPrice) {
      filters.price = {
        ...filters.price,
        $lte: parseFloat(req.query.maxPrice),
      };
    }

    const products = await Product.find(filters).sort({ createdAt: -1 });
    res.send(products);
  } catch (error) {
```

```
    res.status(500).send({ error: error.message });
  }
});
```

```
app.get("/api/products/:id", async (req, res) => {
  try {
    const product = await Product.findById(req.params.id);
    if (!product) {
      return res.status(404).send({ error: "Product not found" });
    }
    res.send(product);
  } catch (error) {
    res.status(500).send({ error: error.message });
  }
});
```

```
app.put("/api/products/:id", async (req, res) => {
  try {
    const product = await Product.findByIdAndUpdate(req.params.id, req.body, {
      new: true,
      runValidators: true,
    });
    if (!product) {
      return res.status(404).send({ error: "Product not found" });
    }
    res.send(product);
  } catch (error) {
    res.status(400).send({ error: error.message });
  }
});
```

```
app.delete("/api/products/:id", async (req, res) => {
  try {
    const product = await Product.findByIdAndDelete(req.params.id);
    if (!product) {
      return res.status(404).send({ error: "Product not found" });
    }
    res.send(product);
  } catch (error) {
    res.status(500).send({ error: error.message });
  }
});
```

```
app.get("/api/search", async (req, res) => {
  try {
    const searchQuery = req.query.q;
```

```

const products = await Product.find({
  $or: [
    { title: { $regex: searchQuery, $options: "i" } },
    { description: { $regex: searchQuery, $options: "i" } },
  ],
}).sort({ createdAt: -1 });

res.send(products);
} catch (error) {
  res.status(500).send({ error: error.message });
}
});

app.get("/api/categories", async (req, res) => {
  try {
    const categories = await Product.distinct("category");
    res.send(categories);
  } catch (error) {
    res.status(500).send({ error: error.message });
  }
});

const port = process.env.PORT || 3000;
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});

```

IMPORT THIS FILE "data.json" FROM MONGODB

```

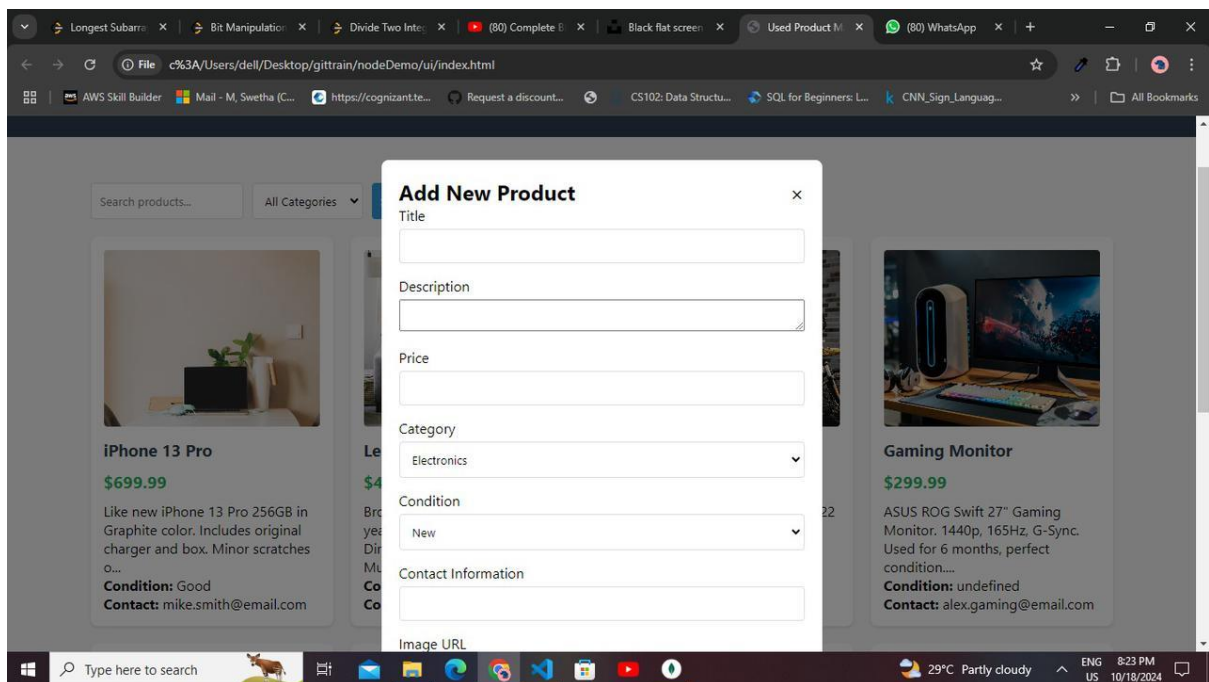
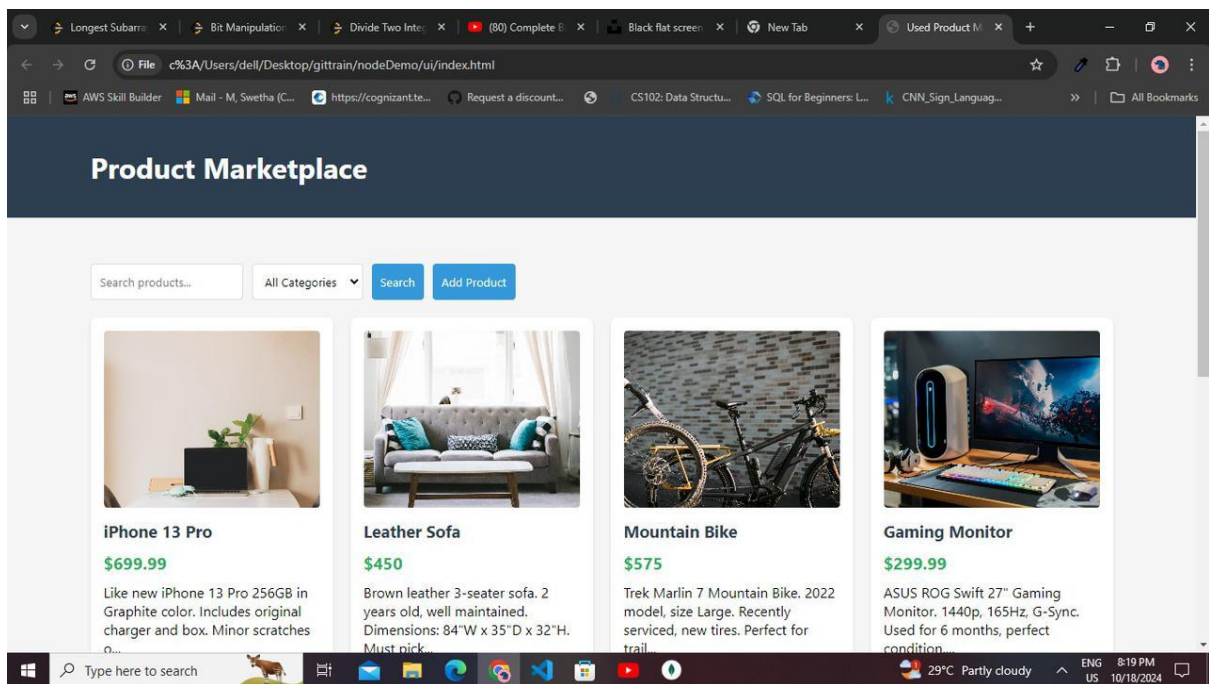
-----MONGODB DATA "data.json" file-----
[
  {
    "title": "iPhone 13 Pro",
    "description": "Like new iPhone 13 Pro 256GB in Graphite color. Includes original charger and box. Minor scratches on screen protector only.",
    "price": 699.99,
    "category": "Electronics",
    "contact": "mike.smith@email.com",
    "image": "https://example.com/iphone13.jpg",
    "createdAt": "2024-10-18T08:30:00.000Z",
    "condition": "Good"
  },
  {
    "title": "Leather Sofa",

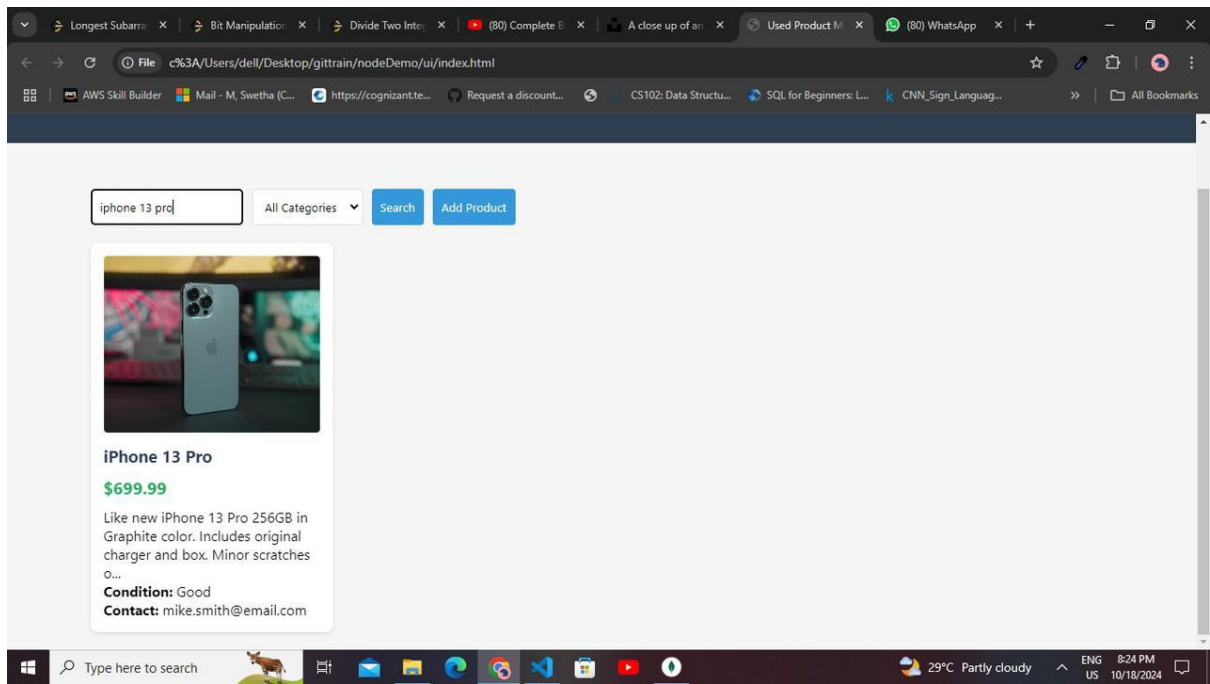
```



```
    "description": "Brown leather 3-seater sofa. 2 years old, well maintained. Dimensions: 84\"W x 35\"D x 32\"H. Must pick up.",
    "price": 450.0,
    "category": "Furniture",
    "contact": "sarah.jones@email.com",
    "image": "https://example.com/sofa.jpg",
    "createdAt": "2024-10-17T15:45:00.000Z",
    "condition": "New"
  },
  {
    "title": "Mountain Bike",
    "description": "Trek Marlin 7 Mountain Bike. 2022 model, size Large. Recently serviced, new tires. Perfect for trails.",
    "price": 575.0,
    "category": "Sports",
    "contact": "john.doe@email.com",
    "image": "https://example.com/bike.jpg",
    "createdAt": "2024-10-17T12:15:00.000Z",
    "condition": "Good"
  },
  {
    "title": "Programming Books Bundle",
    "description": "Collection of 5 JavaScript books including 'Eloquent JavaScript' and 'You Don't Know JS'. All in excellent condition.",
    "price": 85.0,
    "category": "Books",
    "contact": "emma.wilson@email.com",
    "image": "https://example.com/books.jpg",
    "createdAt": "2024-10-16T09:20:00.000Z",
    "condition": "New"
  }
}]
```

OUTPUT:





RESULT:

Ex. No: 6	DEVELOP A LEAVE MANAGEMENT SYSTEM FOR AN ORGANIZATION WHERE USERS CAN APPLY DIFFERENT TYPES OF LEAVES SUCH AS CASUAL LEAVE AND MEDICAL LEAVE. THEY ALSO CAN VIEW THE AVAILABLE NUMBER OF DAYS.

AIM:

ALGORITHM:

PROGRAM:
BACKEND (Node.js & Express.js):
server.js:

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const app = express();
// Middleware
app.use(bodyParser.json());
// Database connection
mongoose.connect('mongodb://localhost/leave_management', { useNewUrlParser:
true,
useUnifiedTopology: true })
.then(() => console.log('Connected to MongoDB...'))
.catch(err => console.error('Could not connect to MongoDB...', err));
// Leave Schema
const leaveSchema = new mongoose.Schema({
  user: String,
  type: String,
  days: Number,
  status: String // pending, approved, rejected
});
const Leave = mongoose.model('Leave', leaveSchema);
// Routes
app.post('/apply-leave', async (req, res) => {
  const { user, type, days } = req.body;
  const newLeave = new Leave({
    user,
    type,
    days,
    status: 'pending'
  });
  await newLeave.save();
  res.send("Leave application submitted!");
});
app.get('/view-leave/:user', async (req, res) => {
  const leaves = await Leave.find({ user: req.params.user });
  res.send(leaves);
});
// Start Server
app.listen(3000, () => console.log('Server started on port 3000'));
```

FRONTEND (HTML/CSS):

index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Leave Management System</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="container">
    <h1>Leave Management System</h1>
    <div class="apply-leave">
      <h2>Apply for Leave</h2>
      <form id="leaveForm">
        <label for="user">Employee Name:</label>
        <input type="text" id="user" name="user" required>
        <label for="type">Leave Type:</label>
        <select id="type" name="type">
          <option value="casual">Casual Leave</option>
          <option value="medical">Medical Leave</option>
        </select>
        <label for="days">Number of Days:</label>
        <input type="number" id="days" name="days" required>
        <button type="submit">Apply</button>
      </form>
    </div>
    <div class="view-leaves">
      <h2>Your Leaves</h2>
      <button onclick="viewLeaves()">View Leaves</button>
      <div id="leaveDetails"></div>
    </div>
  </div>
  <script src="app.js"></script>
</body>
</html>
```

CSS (style.css):

```
body {
  font-family: Arial, sans-serif;
  background-color: #f4f4f4;
  padding: 20px;
}
.container {
  width: 50%;
  margin: auto;
  background: white;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
h1, h2 {
  text-align: center;
  color: #333;
}
form {
  display: flex;
  flex-direction: column;
}
input, select {
  padding: 10px;
  margin: 5px 0;
  border-radius: 5px;
  border: 1px solid #ddd;
}
button {
  padding: 10px;
  background-color: #28a745;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}
button:hover {
  background-color: #218838;
}
```

FRONTEND JAVASCRIPT (Client-Side Logic):

app.js:

```
// Apply Leave
document.getElementById('leaveForm').addEventListener('submit', async function(e) {
  e.preventDefault();
  const user = document.getElementById('user').value;
  const type = document.getElementById('type').value;
  const days = document.getElementById('days').value;
  const response = await fetch('/apply-leave', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ user, type, days })
  });
  const result = await response.text();
  alert(result);
});

// View Leaves
async function viewLeaves() {
  const user = prompt('Enter your name:');
  const response = await fetch(`/view-leave/${user}`);
  const leaves = await response.json();
  let output = '<h3>Leave Details</h3><ul>';
  leaves.forEach(leave => {
    output += `<li>${leave.type}: ${leave.days} days (${leave.status})</li>`;
  });
  output += '</ul>';
  document.getElementById('leaveDetails').innerHTML = output;
}
```


OUTPUT:

Leave Management System

Apply for Leave

Employee Name:

Leave Type:

Casual Leave

Casual Leave

Medical Leave

Apply

Your Leaves

View Leaves

Leave Management System

Apply for Leave

Employee Name:

Leave Type:

Casual Leave

Number of Days:

Apply

Your Leaves

View Leaves

RESULTT:

Ex. No: 7	DEVELOP A SIMPLE DASHBOARD FOR PROJECT MANAGEMENT WHERE THE STATUSES OF VARIOUS TASKS ARE AVAILABLE. NEW TASKS CAN BE ADDED AND THE STATUS OF EXISTING TASKS CAN BE CHANGED AMONG PENDING. IN PROGRESS OR COMPLETED.

AIM:

ALGORITHM:

PROGRAM:

BACKEND (Node.js & Express.js):

Server.js:

```
Const express = require('express');
Const bodyParser = require('body-parser');
Const app = express();
App.use(bodyParser.json());
App.use(express.static('public'));
Let tasks = [];
// Fetch all tasks
App.get('/tasks', (req, res) => {
  Res.json(tasks);
});
// Add a new task
App.post('/tasks', (req, res) => {
  Const { name } = req.body;
  Const newTask = { id: tasks.length + 1, name, status: 'Pending' };
  Tasks.push(newTask);
  Res.json(newTask);
});
// Update task status
App.put('/tasks/:id', (req, res) => {
  Const { id } = req.params;
  Const { status } = req.body;
  Const task = tasks.find(t => t.id === parseInt(id));
  If (!task) return res.status(404).send("Task not found.");
  Task.status = status;
  Res.json(task);
});
App.listen(3000, () => console.log("Server started on port 3000"));
```

FRONTEND (HTML/CSS/JavaScript):

Index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Project Management Dashboard</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="container">
```

```
<h1>Project Management Dashboard</h1>
<div class="task-form">
<h2>Add New Task</h2>
<input type="text" id="taskName" placeholder="Enter task name" required>
<button id="addTaskBtn">Add Task</button>
</div>
<div class="task-list">
<h2>Task List</h2>
<ul id="tasks"></ul>
</div>
</div>
<script src="app.js"></script>
</body>
</html>
```

Style.css

Body {

Font-family: Arial, sans-serif;

Background-color: #f9f9f9;

Margin: 0;

Padding: 0;

}

.container {

Width: 60%;

Margin: auto;

Padding: 20px;

Background-color: white;

Box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

Border-radius: 10px;

Margin-top: 30px;

}

H1, h2 {

Text-align: center;

Color: #333;

}

.task-form {

Display: flex;

Justify-content: space-between;

Margin-bottom: 20px;

}

Input {

Width: 80%;

Padding: 10px;

Font-size: 16px;

Border-radius: 5px;

Border: 1px solid #ccc;

}

```

Button {
  Padding: 10px;
  Background-color: #28a745;
  Color: white;
  Border: none;
  Border-radius: 5px;
  Cursor: pointer;
}
Button:hover {
  Background-color: #218838;
}
Ul {
  List-style-type: none;
  Padding: 0;
}
Li {
  Padding: 10px;
  Border-bottom: 1px solid #ddd;
  Display: flex;
  Justify-content: space-between;
  Align-items: center;
}
.status-dropdown {
  Margin-left: 20px;
}

```

FRONTEND JAVASCRIPT (Client-Side Logic):

App.js:

```

Document.addEventListener('DOMContentLoaded', () => {
  // Fetch and display tasks
  fetchTasks();
  // Add new task
  Document.getElementById('addTaskBtn').addEventListener('click', async () => {
    Const taskName = document.getElementById('taskName').value;
    If (taskName) {
      Await fetch('/tasks', {
        Method: 'POST',
        Headers: { 'Content-Type': 'application/json' },
        Body: JSON.stringify({ name: taskName })
      });
      Document.getElementById('taskName').value = '';
      fetchTasks();
    }
  });
});

```

```

});
// Fetch tasks and display them
Async function fetchTasks() {
  Const response = await fetch('/tasks');
  Const tasks = await response.json();
  Const taskList = document.getElementById('tasks');
  taskList.innerHTML = "";
  tasks.forEach(task => {
    const listItem = document.createElement('li');
    listItem.innerHTML = `
    <span>${task.name}</span>
    <select class="status-dropdown" data-id="${task.id}">
    <option value="Pending" ${task.status === 'Pending' ? 'selected' : ""}>Pending</option>
    <option value="In Progress" ${task.status === 'In Progress' ? 'selected' : ""}>In
    Progress</option>
    <option value="Completed" ${task.status === 'Completed' ? 'selected' :
    ""}>Completed</option>
    </select>
    `;
    taskList.appendChild(listItem);
  });
  // Add event listeners to dropdowns for updating status
  Document.querySelectorAll('.status-dropdown').forEach(dropdown => {
    Dropdown.addEventListener('change', async e => {
      Const id = e.target.getAttribute('data-id');
      Const status = e.target.value;
      Await fetch(`/tasks/${id}`, {
        Method: 'PUT',
        Headers: { 'Content-Type': 'application/json' },
        Body: JSON.stringify({ status })
      });
    });
  });
  fetchTasks();
});
}

```

OUTPUT:

Project Management Dashboard

Add New Task

Enter task name

Add Task

Task List

RESULT:

Ex. No: 8	DEVELOP AN ONLINE SURVEY APPLICATION WHERE A COLLECTION OF QUESTIONS IS AVAILABLE AND USERS ARE ASKED TO ANSWER ANY RANDOM 5 QUESTIONS

AIM:

ALGORITHM:

PROGRAM:

```
<!DOCTYPE.html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Temperature Conversion</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f9;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      margin: 0;
    }

    .container {
      background-color: white;
      border-radius: 10px;
      padding: 20px;
      box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
      text-align: center;
      width: 300px;
    }

    h1 {
      color: #4CAF50;
      margin-bottom: 20px;
    }

    .input-group {
      margin-bottom: 15px;
      text-align: left;
    }

    label {
      display: block;
      font-size: 16px;
      margin-bottom: 5px;
    }

    input[type="number"],
```

```
select {
  width: 100%;
  padding: 10px;
  font-size: 16px;
  margin-bottom: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
}

button.convert-btn {
  background-color: #4CAF50;
  color: white;
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  font-size: 16px;
  width: 100%;
}

button.convert-btn:hover {
  background-color: #45a049;
}

#result {
  margin-top: 20px;
  font-size: 18px;
  color: #333;
}
</style>
</head>
<body>
<div class="container">
  <h1>Temperature Converter</h1>
  <div class="input-group">
    <label for="tempInput">Enter Temperature:</label>
    <input type="number" id="tempInput" placeholder="Temperature">
  </div>

  <div class="input-group">
    <label for="unitSelect">Select Conversion:</label>
    <select id="unitSelect">
      <option value="Celsius">Celsius to Fahrenheit</option>
      <option value="Fahrenheit">Fahrenheit to Celsius</option>
    </select>
  </div>
</div>
```

```
<button class="convert-btn" onclick="convertTemperature()">Convert</button>

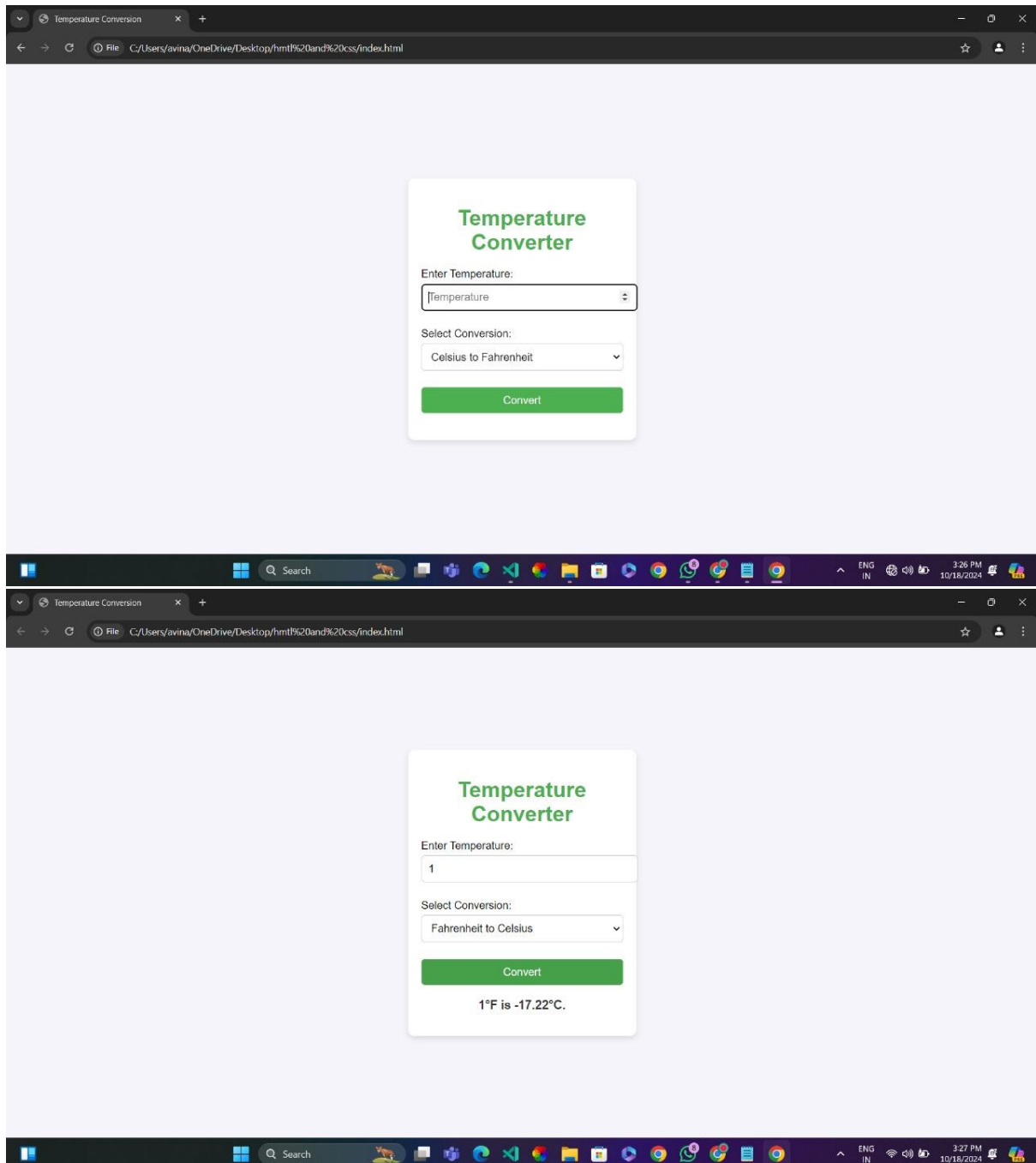
<h2 id="result"></h2>
</div>

<script>
function convertTemperature() {
  const tempInput = document.getElementById('tempInput').value;
  const unitSelect = document.getElementById('unitSelect').value;
  const resultElement = document.getElementById('result');

  let result;
  if (!tempInput) {
    resultElement.textContent = 'Please enter a temperature.';
    return;
  }

  if (unitSelect === 'Celsius') {
    result = (tempInput * 9/5) + 32; // Celsius to Fahrenheit
    resultElement.textContent = `${tempInput}°C is ${result.toFixed(2)}°F;`
  } else {
    result = (tempInput - 32) * 5/9; // Fahrenheit to Celsius
    resultElement.textContent = `${tempInput}°F is ${result.toFixed(2)}°C.;`
  }
}
</script>
</body>
</html>
```

OUTPUT:



RESULT:

Ex. No: 9	CREATE YOUR OWN WEB APPLICATION

AIM:

ALGORITHM:

PROGRAM:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Survey App</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f9;
      color: #333;
      padding: 20px;
      margin: 0;
    }
    h1 {
      text-align: center;
      color: #4CAF50;
    }
    button {
      background-color: #4CAF50;
      color: white;
      padding: 10px 20px;
      border: none;
      border-radius: 5px;
      cursor: pointer;
      font-size: 16px;
    }
    button:hover {
      background-color: #45a049;
    }
    form {
      margin-top: 20px;
    }
    .question {
      background-color: #fff;
      border: 1px solid #ccc;
      border-radius: 5px;
      padding: 15px;
      margin-bottom: 20px;
      box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    }
    label {
      font-size: 16px;
```

```

    font-weight: bold;
    color: #333;
  }
  input[type="radio"],
  input[type="text"] {
    margin: 10px 0;
  }
  input[type="radio"] {
    margin-right: 5px;
  }
  input[type="text"] {
    width: 100%;
    padding: 8px;
    box-sizing: border-box;
    border: 1px solid #ccc;
    border-radius: 5px;
  }
  #submit-btn {
    display: block;
    width: 30%;
    background-color: #4CAF50;
    color: white;
    padding: 10px;
    border: none;
    border-radius: 5px;
    font-size: 16px;
    cursor: pointer;
    margin: 20px auto 0; /* Center the button */
    text-align: center;
  }
  #submit-btn:hover {
    background-color: #45a049;
  }
</style>
</head>
<body>
  <h1>Survey Application</h1>
  <button id="fetch-questions">Get Random Questions</button>
  <form id="survey-form"></form>
  <button id="submit-btn" type="submit" form="survey-form">Submit
Responses</button>

  <script>
    document.getElementById('fetch-questions').addEventListener('click', async () => {
      const response = await fetch('http://localhost:3000/questions/random');
      const questions = await response.json();

```



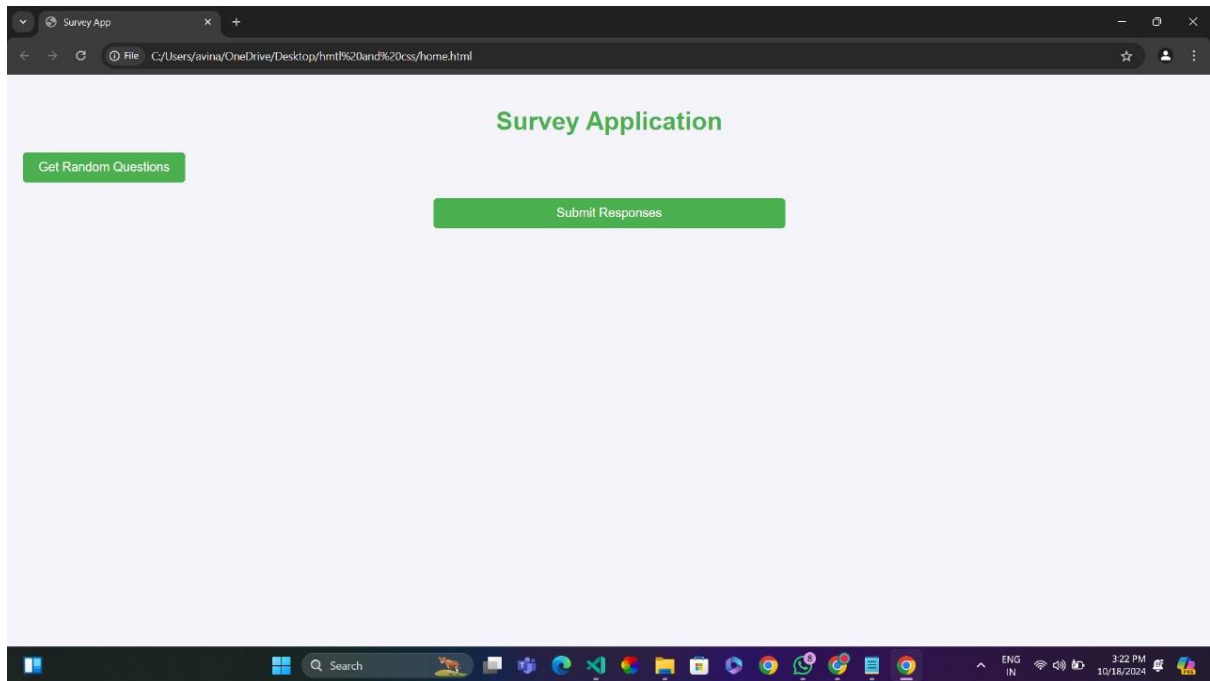
```

const form = document.getElementById('survey-form');
form.innerHTML = ''; // Clear previous questions
questions.forEach(q => {
  const questionElement = document.createElement('div');
  questionElement.className = 'question';
  questionElement.innerHTML = <label>${q.question_text}</label><br>;
  if (q.question_type === 'multiple-choice') {
    q.options.forEach(option => {
      questionElement.innerHTML += <input type="radio" name="${q._id}"
value="${option}"> ${option}<br>;
    });
  } else {
    questionElement.innerHTML += <input type="text" name="${q._id}"><br>;
  }
  form.appendChild(questionElement);
});
});

document.getElementById('survey-form').addEventListener('submit', async (event)
=> {
  event.preventDefault();
  const formData = new FormData(event.target);
  const responses = [];
  formData.forEach((value, key) => {
    responses.push({ question_id: key, response: value });
  });
  await fetch('http://localhost:3000/responses', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(responses),
  });
  alert('Responses submitted!');
});
</script>
</body>
</html>

```

OUTPUT:



RESULT: