Experiment-1

AIM: To Study about CSS and the various properties used for styling.

ABOUT CSS:

Cascading Style Sheets (CSS) is a powerful language that defines the presentation and layout of web pages. With CSS, developers can customize the appearance of HTML elements, controlling everything from colours and fonts to spacing and positioning. By separating the content (HTML) from its presentation (CSS), CSS enables developers to create visually stunning and responsive websites that adapt to different devices and screen sizes. CSS employs selectors to target specific elements and apply styles, offering flexibility and granularity in design. Additionally, CSS features such as cascading, inheritance, and specificity provide mechanisms for managing and organizing styles efficiently. With its intuitive syntax and extensive capabilities, CSS plays a crucial role in shaping the visual identity and user experience of modern web applications, making it an indispensable tool for web developers worldwide.

1. Colour:

Colour: Sets the text colour.
background-colour: Sets the background colour.
opacity: Sets the opacity of an element.

2. Typography:

font-family: Specifies the font family for text.
font-size: Sets the size of the font.
font-weight: Sets the thickness of the font.
text-align: Aligns text horizontally.

3. Layout:

display: Specifies the display behavior of an element.
width and height: Sets the width and height of an element.
margin, padding: Controls the spacing around an element.
float: Positions an element to the left or right of its container.

4. Positioning:

position: Sets the positioning method for an element.
top, bottom, left, right: Positions an element relative to its containing element.
z-index: Specifies the stack order of an element.

5. Border:

border: Sets the border properties (width, style, colour).
border-radius: Rounds the corners of an element.

6. Background:

background-image: Sets the background image.
background-size: Specifies the size of the background image.
background-repeat: Controls how the background image is repeated.

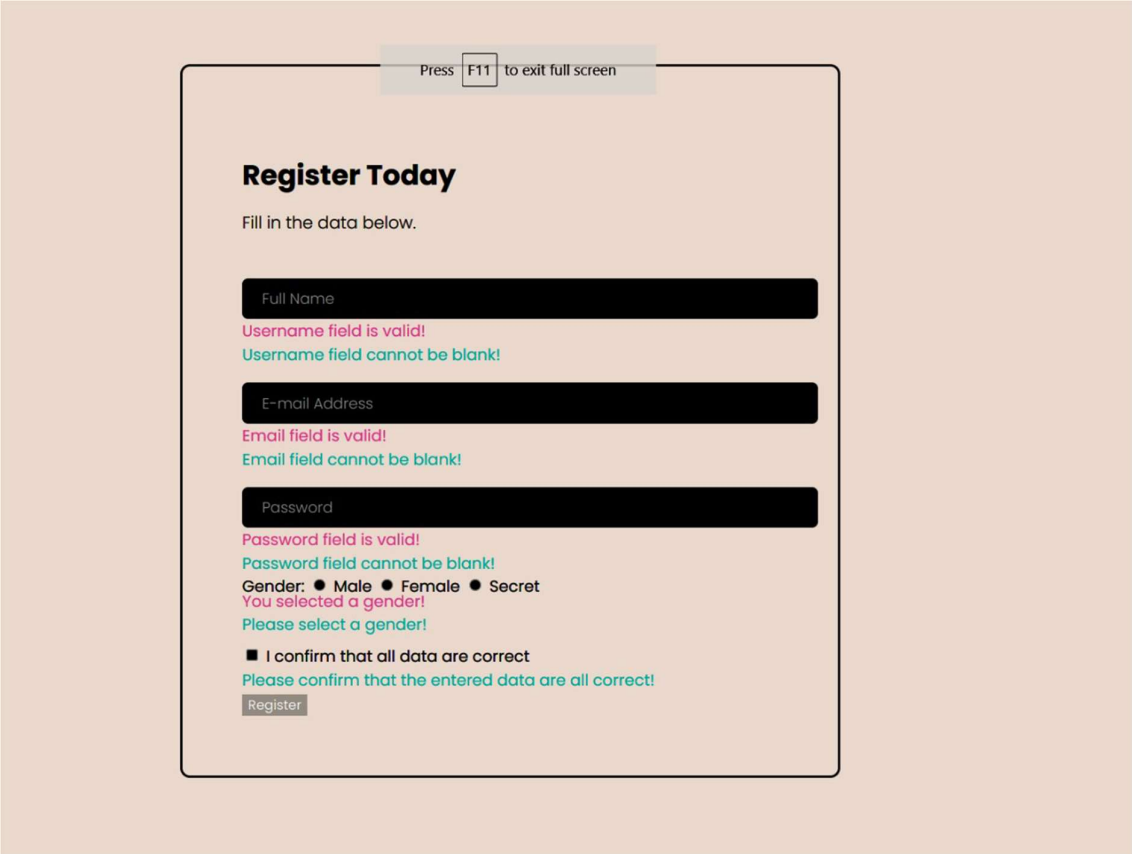
7. Animations and Transitions:

animation: Specifies the keyframes for an animation.
transition: Specifies the transition properties for an element.

8. Flexbox:

display: flex: Enables the flexbox layout.
flex-direction: Specifies the direction of the flex container.
justify-content, align-items: Aligns items within a flex container.

CREATED USING HTML AND CSS:



CODE SNIPPETS:

```
.btn-primary {
    background-color: #6C757D;
    outline: none;
    border: 0px;
    box-shadow: none;
}

.btn-primary:hover,
.btn-primary:focus,
.btn-primary:active {
    background-color: #495056;
    outline: none !important;
    border: none !important;
    box-shadow: none;
}

.form-content textarea {
    position: static !important;
    width: 100%;
    padding: 8px 20px;
    border-radius: 6px;
    text-align: left;
    background-color: #fff;
    border: 0;
    font-size: 15px;
    font-weight: 300;
    color: #8D8D8D;
    outline: none;
    resize: none;
    height: 120px;
    -webkit-transition: none;
    transition: none;
    margin-bottom: 14px;
}
```

```
.form-content p {
    color: #fff;
    text-align: left;
    font-size: 17px;
    font-weight: 300;
    line-height: 20px;
    margin-bottom: 30px;
}


.form-content label,
.was-validated .form-check-input:invalid~.form-check-label,
.was-validated .form-check-input:valid~.form-check-label {
    color: #fff;
}

.form-content input[type=text],
.form-content input[type=password],
.form-content input[type=email],
.form-content select {
    width: 100%;
    padding: 9px 20px;
    text-align: left;
    border: 0;
    outline: 0;
    border-radius: 6px;
    background-color: #fff;
    font-size: 15px;
    font-weight: 300;
    color: #8D8D8D;
    -webkit-transition: all 0.3s ease;
    transition: all 0.3s ease;
    margin-top: 16px;
}
```

Experiment-2

AIM: Task management tool: Login/Register to the application, add daily tasks, Assign a due date of completion, Mark them as complete/incomplete and View weekly/monthly statistics of their to-dos.

DESCRIPTION:

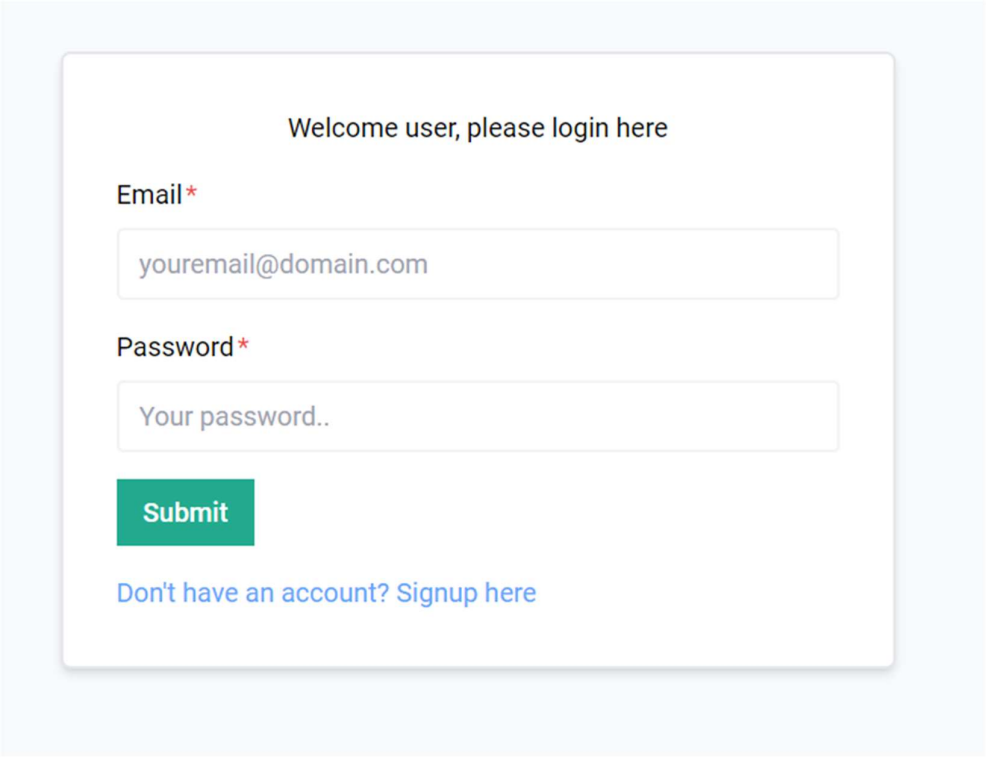
A basic task management application, the MERN (MongoDB, Express.js, React.js, Node.js) stack forms the core technology stack. MongoDB stores user data, while Express.js facilitates backend server creation for tasks like user authentication. Node.js powers backend logic, handling HTTP requests efficiently. On the frontend, React.js creates interactive user interfaces, rendering login forms, task input fields, and statistical views. This seamless integration ensures a cohesive development process for building a responsive task management tool.

TOOLS AND TECHNOLOGIES USED:

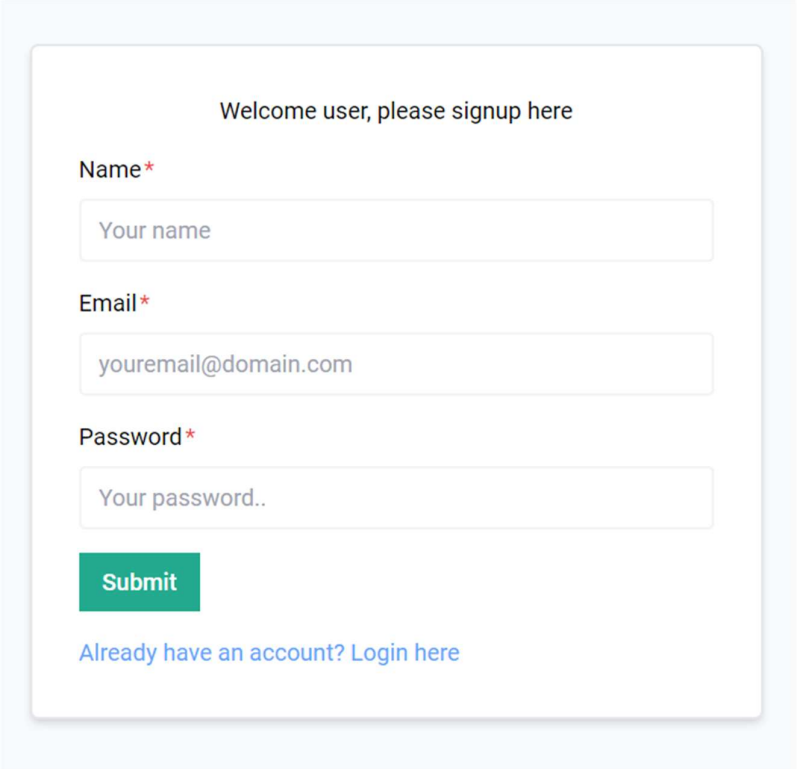
1)VS CODE
2) HTML,CSS,JAVASCRIPT,FIREBASE

WEBPAGE:

SIGN IN FUNCTIONALITY:



SIGN UP FUNCTIONALITY:



TASK MANAGEMENT HOMEPAGE:

CODE SNIPPETS:

HTML AND CSS:

```
const Home = () => {
  return (
    <>
      <nav className="relative flex items-center justify-around h-14 bg-blue-100 text-black px-4">
        <div className="mt-10">
          <h1 className="text-4xl font-bold">Task Manager</h1>
        </div>
      </nav>
      <div className="bg-blue-100 min-h-screen px-4 py-8">
        <form
          className="flex items-center justify-center mb-8"
          onSubmit={addTask}
        >
          <input
            type="text"
            className="mr-4 p-2 rounded-lg border-2 border-blue-500"
            placeholder="Enter task"
            value={task}
            onChange={(e) => setTask(e.target.value)}
          />
          <input
            type="date"
            className="mr-4 p-2 rounded-lg border-2 border-blue-500"
            value={dueDate}
            onChange={(e) => setDueDate(e.target.value)}
          />
          <button className="bg-blue-600 p-2 rounded-lg text-white hover:bg-blue-700 transition-colors">
            Add Task
          </button>

          <button
            onClick={handleSub}
            className="bg-red-500 p-2 ml-2 rounded-lg text-white "
          >
            Logout
          </button>
        </form>
        <ul>
          {tasks.map((task) => (
            <li
              key={task.id}
              className="flex items-center justify-between bg-white rounded-lg shadow-md p-4 mb-4"
```

REACT:

```
import React, { useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import { auth } from "../Firebase/Firebase";
import { signInWithEmailAndPassword } from "firebase/auth";

const Login = () => {
  const [values, setValues] = useState({
    email: "",
    password: "",
  });

  const Navigate = useNavigate();
  const [errormsg, setErrormsg] = useState("");
  const [SubmitButtonDisabled, setSubmitButtonDisabled] = useState(false);

  const handleSubmit = () => {
    if (!values.email || !values.password) {
      setErrormsg("Fill all fields");
      return;
    }
    setErrormsg("");
    setSubmitButtonDisabled(true);
    signInWithEmailAndPassword(auth, values.email, values.password)
      .then(async (res) => {
        setSubmitButtonDisabled(false);
        alert("Succesfully Signed up");

        Navigate("/Home");
        // console.log(user);
      })
      .catch((err) => {
        setSubmitButtonDisabled(false);
        setErrormsg(err.message);
      });
  };
```

AUTHENTICATION:

```
import { initializeApp } from "firebase/app";
import { getAuth } from "firebase/auth";

const firebaseConfig = {

};

const app = initializeApp(firebaseConfig);

const auth = getAuth();
export { app, auth };
```

Experiment-3

AIM: To create an interactive blogging platform using MERN STACK.

DESCRIPTION:

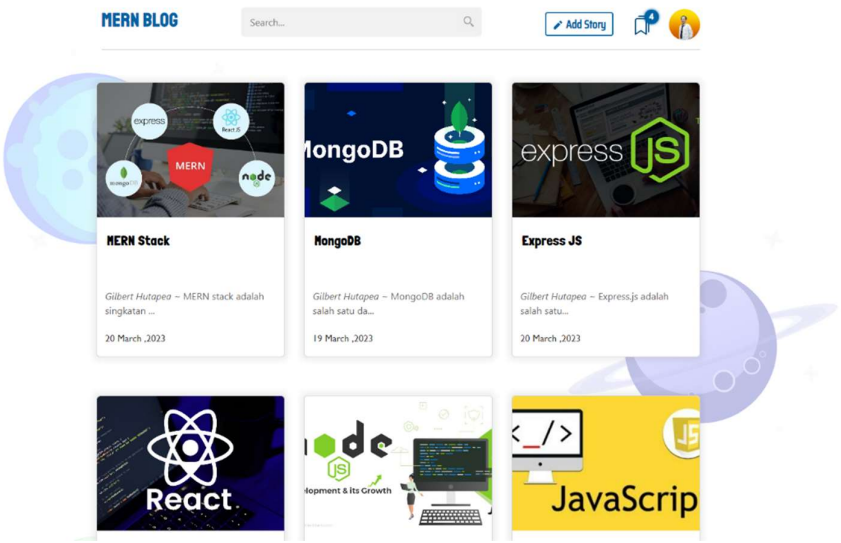
A blogging platform, that uses MERN (MongoDB, Express.js, React.js, Node.js) stack is utilized to create a dynamic and user-friendly interface. MongoDB acts as the database for storing blog posts and user data, offering flexibility and scalability. Express.js manages server-side logic and routing, ensuring smooth communication between frontend and backend components. Node.js powers the backend, handling operations like user authentication and post creation. On the frontend, React.js enables the creation of interactive user interfaces, rendering blog posts and allowing users to add new content. Together, the MERN stack provides a foundation for building a simple yet functional blogging platform, allowing users to share their thoughts and ideas with others.

TOOLS AND TECHNOLOGIES USED:

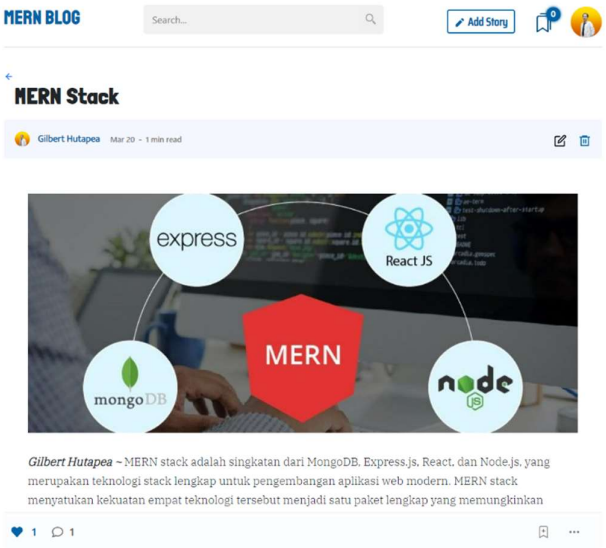
1)VS CODE
2) REACT,TAILWIND

WEBPAGE

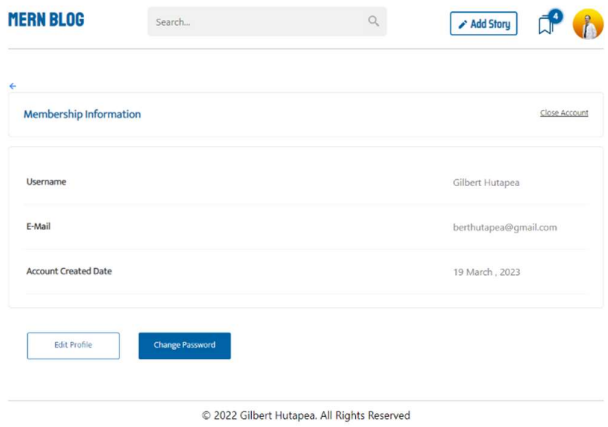
HOME PAGE:



VIEW BLOG



UPDATE ACCOUNT



CODE SNIPPETS:

CREATED USING REACT

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import { ContextProvider } from './components/Context/Context';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <ContextProvider>
      <App />
    </ContextProvider>
  </React.StrictMode>
);
```

CONTROLLER FOR CREATING AND UPDATING POST

```
import './write.css'
import { useState, useContext } from "react"
import axios from "axios"
import { Context } from './Context/Context'

const Write = () => {
    const [title, settitle] = useState("")
    const [desc, setdesc] = useState("")
    const [file, setfile] = useState(null)
    const { user } = useContext(Context)
    const handlesubmit = async (e) => {
        e.preventDefault();
        const newPost = {
            username: user.username,
            title,
            desc,
        }
        if (file) {
            const data = new FormData();
            const filename = Date.now() + file.name;
            data.append("name", filename)
            data.append("file", file)
            newPost.photo = filename;
            try {
                await axios.post("/upload", data)
                console.log(data)
            } catch (err) {

            }
        }
        try {
            const res = await axios.post("/posts", newPost)
            window.location.replace("/post/" + res.data._id)
        } catch (err) {

        }
    }
```

```
const SinglePost = () => {
    const handleclick = async () => {
            window.location.replace("/")

        } catch (err) {

        }
    }
    return (
        <div className='singlepost'>
            <div className="singlepostwrapper">
                {post.photo &&

                    <img src={PF + post.photo} alt="" className="singlepostimg" />
                }
                {
                    updatemode ? <input
                        type="text"
                        value={title}
                        className='singleposttitleinput'
                        autoFocus
                        onChange={(e) => settitle(e.target.value)} /> : (

                        <h1 className="singleposttitle">
                            {title}
                            {post.username === user?.username && (
                                <div className="singlepostedit">
                                    <i className="singleposticon fa-regular fa-pen-to-square"
                                        onClick={() => {
                                            setupdatemode(true)
                                        }}></i>
                                    <i className="singleposticon fa-regular fa-trash-can" onClick={handleclick}></i>
                                </div>
                            )}
                        </h1>
                    )
                }
                <div className="singlepostinfo">
```

Experiment-4

AIM: Crafting an innovative and immersive social media platform with the MERN (MongoDB, Express.js, React.js, Node.js) stack, facilitating seamless interaction, content sharing, and community engagement for users worldwide.

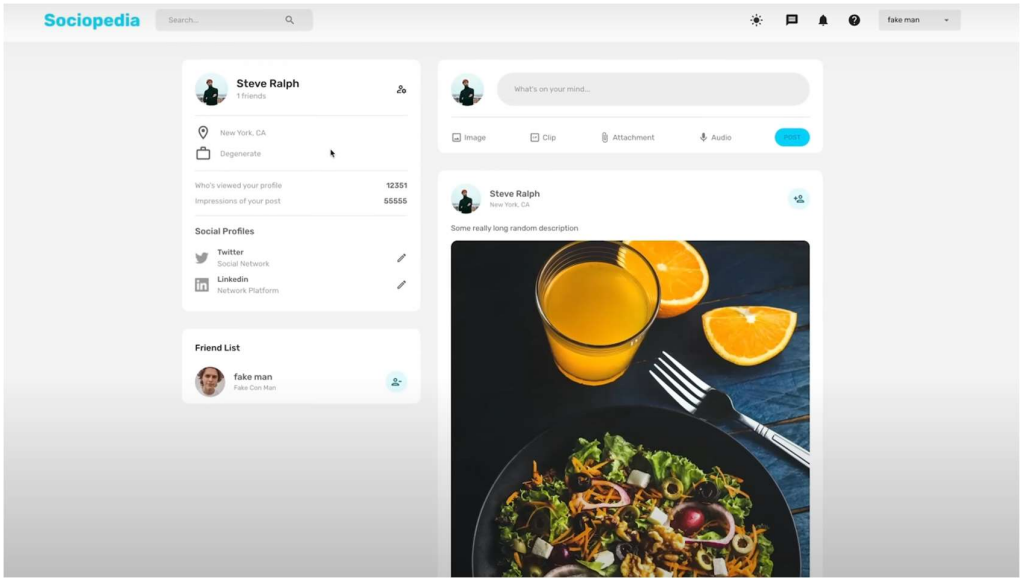
DESCRIPTION:

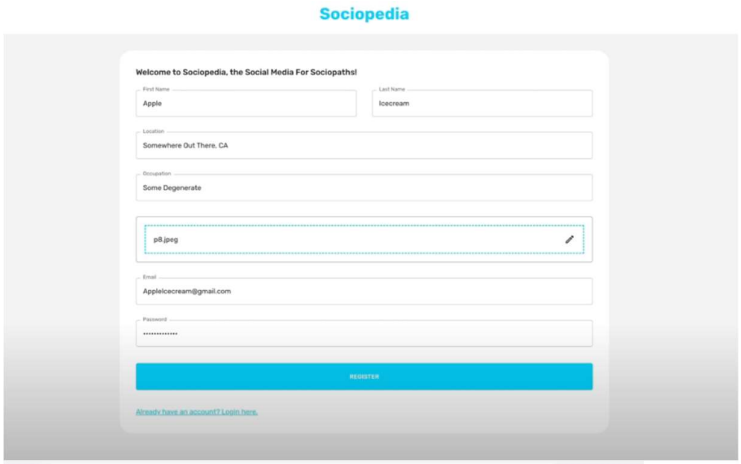
Building a social media platform with the MERN (MongoDB, Express.js, React.js, Node.js) stack offers a dynamic and interactive space for users to connect, share content, and engage with communities. MongoDB serves as the database, storing user profiles, posts, and interactions, providing scalability for growing user bases. Express.js handles server-side logic and routing, ensuring smooth communication between the frontend and backend. Node.js powers the backend, managing user authentication, post creation, and data retrieval. On the frontend, React.js enables the creation of responsive user interfaces, facilitating seamless navigation and interaction. Together, the MERN stack provides a solid foundation for crafting a basic yet innovative social media platform, fostering connections and conversations among users from diverse backgrounds.

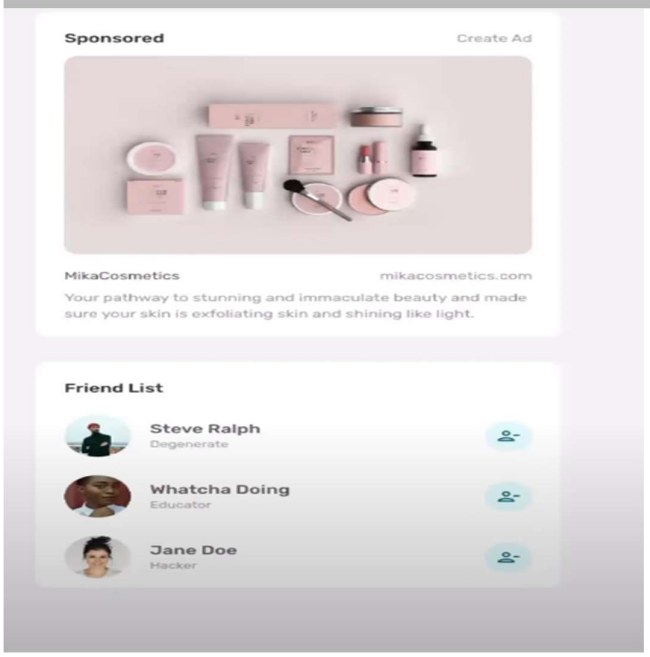
TOOLS AND TECHNOLOGIES USED:

1)VS CODE
2) HTML,TAILWIND,JAVASCRIPT

WEBPAGE







CODE SNIPPETS:

HTML AND TAILWIND:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">


    <title>Discord | Your Place to T Follow link (cmd + click) le>
    <link rel="shortcut icon" href="./images/favicon.ico" type="image/x-icon">

    <link rel="stylesheet" href="main.css">
</head>
<body>

    <!--nav section-->
    <div class="bg-[#404EED] font-ggSans">
        <!-- Navbar -->
        <nav class="text-[#fff]  max-w-[1200px] mx-auto
        flex justify-between h-[72px] items-center xl:px-0 px-7">
          <!-- Logo -->
          <div>
            <img src="./images/logo.svg" alt="">
          </div>
          <!-- Nav Options -->
          <div class="hidden xl:flex gap-5 font-ggSans font-[600] font-[16px]">    'font-[600]' applies the same CSS properties as 'font-[16p
            <a href="#" class="capitalize hover:underline">download</a>
            <a href="#" class="capitalize hover:underline">nitro</a>
            <a href="#" class="capitalize hover:underline">discover</a>
            <a href="#" class="capitalize hover:underline">safety</a>
            <a href="#" class="capitalize hover:underline">support</a>
            <a href="#" class="capitalize hover:underline">blog</a>
            <a href="#" class="capitalize hover:underline">careers</a>
          </div>/.hidden.xl:flex.gap-5.font-ggSans.font-[600].font-[16px]
          <!-- Open DiscordBtn -->
          <div class="hidden xl:flex">
            <a href="#">
              <button class="bg-[#fff] px-4 self-center p-2 text-[#2c2f33]
```

```html
            <p
              worldwide art community. Where just you and a handful of friends can
              spend time together. A place that makes it easy to talk every day
              and hang out more often.
            </p>/.font-[400].text-left.xl:text-center.text-[16px].md:text-[20px].leading-[26px].md:leading-[32.5px].font-[gg.sans].w-[90%].md:

            <div class="hidden xl:flex flex-row item-center justify-center gap-10 font-[500] z-20">
              <a href="#" class="bg-white text-black p-3 px-7 rounded-full hover:text-[#404EED]">
                <p>
                  <i class="fa-solid fa-download mr-2"></i>
                  Download for Windows
                </p>
              </a>/#.bg-white.text-black.p-3.px-7.rounded-full.hover:text-[#.bg-white.text-black.p-3.px-7.rounded-full.hover:text-[#404EED]
              <a href="#" class="bg-[#23272a] text-white p-3 px-7 rounded-full">
                Open Discord in your browser</a>/#.bg-[#.bg-[#23272a].text-white.p-3.px-7.rounded-full
            </div>/.hidden.xl:flex.flex-row.item-center.justify-center.gap-10.font-[500].z-20

            <!-- Mobile/Tablet Mode Start -->
            <div class="flex xl:hidden font-[500] z-20">
              <a href="#" class="bg-white text-black p-3 px-7 rounded-full hover:text-[#404EED]">
                <p>
                  <i class="fa-solid fa-download mr-2"></i>
                  Download from Google Play
                </p>
              </a>/#.bg-white.text-black.p-3.px-7.rounded-full.hover:text-[#.bg-white.text-black.p-3.px-7.rounded-full.hover:text-[#404EED]
            </div>/.flex.xl:hidden.font-[500].z-20

            <img src="./images/Image2.svg" class="block md:hidden ml-[-80px]" alt="" />

            <div class="hidden md:flex xl:hidden mr-[-150px] mt-[-16%] justify-end">
              <img src="./images/Image1.svg" class="w-[70%]" alt="" />
            </div>/.hidden.md:flex.xl:hidden.mr-[-150px].mt-[-16%].justify-end

            <!-- Mobile/Tablet Mode End -->
          </div>/#.text-[#.text-[#fff].container.max-w-[1200px].xl:px-0.px-7.xl:max-w-[900px].mx-auto.text-center.xl:h-[555px].flex.flex-col.g

    </div>/#.bg-[#.bg-[#404EED].-mt-1.relative.w-full.overflow-x-hidden

    <!-- section 1 -->
    <div class="flex container max-w-[1200px] mx-auto
```

```html
          <div class="hidden xl:flex">
            <a href="#">
              <button class="bg-[#fff] px-4 self-center p-2 text-[#2c2f33]
              rounded-full font-[500] text-[14px]">
                Open Discord
              </button>/#.bg-[#.bg-[#fff].px-4.self-center.p-2.text-[#2c2f33].rounded-full.font-[500].text-[14px]
            </a>
          </div>/.hidden.xl:flex

          <!-- Humbarget Option -->
          <div class="xl:hidden block">
            <i class="fa-solid fa-bars text-[23px]"></i>
          </div>/.xl:hidden.block
        </nav>/#.text-[#.text-[#fff].max-w-[1200px].mx-auto.flex.justify-between.h-[72px].items-center.xl:px-0.px-7

    </div>/#.bg-[#.bg-[#404EED].font-ggSans

    <!--hero section-->

    <div class="bg-[#404EED] -mt-1 relative w-full overflow-x-hidden">

          <img src="./images/Image0.svg"
          class="hidden xl:block absolute z-10 bottom-0 top-auto scale-x-125" alt="" />/.hidden.xl:block.absolute.z-10.bottom-0.top-auto.scal
          <img src="./images/Image1.svg" class="hidden xl:block absolute z-10 bottom-0 -right-[5%]" alt="" />
          <img src="./images/Image2.svg" class="hidden xl:block absolute z-10 bottom-0 -left-[5%]" alt="" />
          <div
            class="text-[#fff] container max-w-[1200px] xl:px-0 px-7 xl:max-w-[900px]
            mx-auto text-center xl:h-[555px] flex flex-col gap-5 pt-[50px] xl:pt-[120px]">

            <h2 class="font-[800] text-left xl:text-center text-[34px] md:text-[56px]
            font-['GintoNord'] z-20">
              IMAGINE A PLACE...
            </h2>/.font-[800].text-left.xl:text-center.text-[34px].md:text-[56px].font-[GintoNord].z-20

            <p
              class="font-[400] text-left xl:text-center
              text-[16px] md:text-[20px] leading-[26px]
              md:leading-[32.5px] font-['gg sans'] w-[90%] md:w-[70%] xl:w-[90%] z-20 xl:self-center">
              ...where you can belong to a school club, a gaming group, or a
```

Experiment-5

AIM: To Create a weather app using MERN STACK.

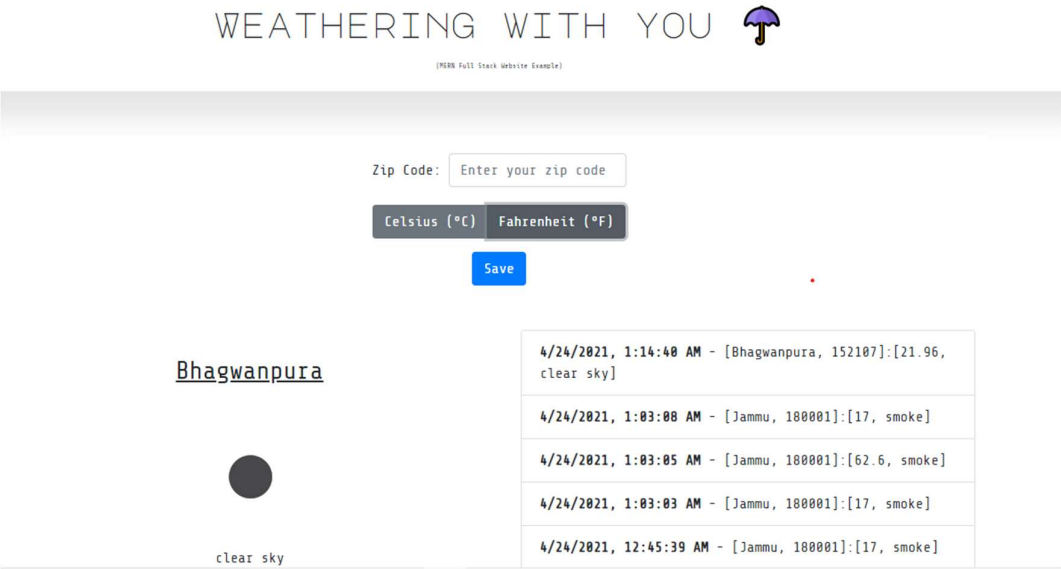
DESCRIPTION:

It is a sleek and intuitive weather application built using HTML, CSS, and JavaScript. Seamlessly blending form and function, it provides users with up-to-date weather information in a visually appealing and easy-to-navigate interface. Users can input their location or allow the app to detect their current location automatically. The app then fetches real-time weather data using APIs, presenting it in a clear and concise manner with detailed forecasts for the upcoming days. With interactive features reflecting current weather conditions and responsive design ensuring optimal viewing on all devices, offers a seamless and enjoyable user experience for staying informed about the weather anytime, anywhere.

TOOLS AND TECHNOLOGIES USED:

1)VS CODE
2) HTML,CSS,JAVASCRIPT

WEBPAGE



HTML CSS AND JS:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>

    <div class="wrapper">

        <h1>Weather App</h1>

        <div class="tab-container">
            <p class="tab" data-userWeather>Your Weather</p>
            <p class="tab" data-searchWeather>Search Weather</p>
        </div>/.tab-container

        <div class="weather-container">

            <!-- grant location container-->
            <div class="grant-location-container">
                <img src="./assets/location.png" width="80" height="80" loading="lazy">
                <p>grant Location Access</p>
                <p>Allow Access to get weather Information</p>
                <button class="btn" data-grantAccess>Grant Access</button>
            </div>/.grant-location-container

            <!-- search form -> form-container-->
            <form class="form-container" data-searchForm>
                <input placeholder="Search for City..." data-searchInput>
                <button class="btn" type="submit">
                    <img src="./assets/search.png"  width="20" height="20" loading="lazy">
                </button>/.btn
            </form>/.form-container

            <!--- loading screen container -->
            <div class="loading-container">
                <img src="./assets/loading.gif" width="150" height="150">
                <p>Loading</p>
            </div>/.loading-container

            <!-- show weather info -->
            <div class="user-info-container">

                <!--city name and Flag-->
                <div class="name">
                    <p data-cityName></p>
                    <img data-countryIcon>
                </div>/.name

                <!-- weather descriptuion-->
                <p data-weatherDesc></p>
                <!--weather Icon-->
                <img data-weatherIcon>
                <!--temperature-->
                <p data-temp></p>

                <!--3 cards - parameters-->
                <div class="parameter-container">

                    <!--card 1-->
                    <div class="parameter">
                        <img src="./assets/wind.png" >
                        <p>windspeed</p>
                        <p data-windspeed></p>
                    </div>/.parameter

                    <!--card 2-->
                    <div class="parameter">
                        <img src="./assets/humidity.png" >
                        <p>humidity</p>
                        <p data-humidity></p>
                    </div>/.parameter

                    <!--card 3-->
                    <div class="parameter">
                        <img src="./assets/cloud.png" >
                        <p>Clouds</p>
                        <p data-cloudiness></p>
                    </div>/.parameter
```

```javascript
function renderWeatherInfo(weatherInfo) {
    //fistly, we have to fethc the elements

    const cityName = document.querySelector("[data-cityName]");
    const countryIcon = document.querySelector("[data-countryIcon]");
    const desc = document.querySelector("[data-weatherDesc]");
    const weatherIcon = document.querySelector("[data-weatherIcon]");
    const temp = document.querySelector("[data-temp]");
    const windspeed = document.querySelector("[data-windspeed]");
    const humidity = document.querySelector("[data-humidity]");
    const cloudiness = document.querySelector("[data-cloudiness]");

    //fetch values from weatherINfo object and put it UI elements
    cityName.innerText = weatherInfo?.name;
    countryIcon.src = `https://flagcdn.com/144x108/${weatherInfo?.sys?.country.toLowerCase()}.png`;
    desc.innerText = weatherInfo?.weather?.[0]?.description;
    weatherIcon.src = `http://openweathermap.org/img/w/${weatherInfo?.weather?.[0]?.icon}.png`;
    temp.innerText = weatherInfo?.main?.temp;
    windspeed.innertext = weatherInfo?.wind?.speed;
    humidity.innertext = weatherInfo?.main?.humidity;
    cloudiness.innerText = weatherInfo?.clouds?.all;


}

function getLocation() {
    if(navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(showPosition);
    }
    else {
        //HW - show an alert for no gelolocation support available
    }
}

function showPosition(position) {

    const userCoordinates = {
        lat: position.coords.latitude,
        lon: position.coords.longitude,
    }

    sessionStorage.setItem("user-coordinates", JSON.stringify(userCoordinates));
    fetchUserWeatherInfo(userCoordinates);
```

```javascript
searchTab.addEventListener("click", () => {
    //pass clicked tab as input paramter
    switchTab(searchTab);
});

//check if cordinates are already present in session storage
function getfromSessionStorage() {
    const localCoordinates = sessionStorage.getItem("user-coordinates");
    if(!localCoordinates) {
        //agar local coordinates nahi mile
        grantAccessContainer.classList.add("active");
    }
    else {
        const coordinates = JSON.parse(localCoordinates);
        fetchUserWeatherInfo(coordinates);
    }

}

async function fetchUserWeatherInfo(coordinates) {
    const {lat, lon} = coordinates;
    // make grantcontainer invisible
    grantAccessContainer.classList.remove("active");
    //make loader visible
    loadingScreen.classList.add("active");

    //API CALL
    try {
        const response = await fetch(
            `https://api.openweathermap.org/data/2.5/weather?lat=${lat}&lon=${lon}&appid=${API_KEY}&units=metric`
          );
        const  data = await response.json();

        loadingScreen.classList.remove("active");
        userInfoContainer.classList.add("active");
        renderWeatherInfo(data);
    }
    catch(err) {
        loadingScreen.classList.remove("active");
        //HW

    }
```

Experiment-6

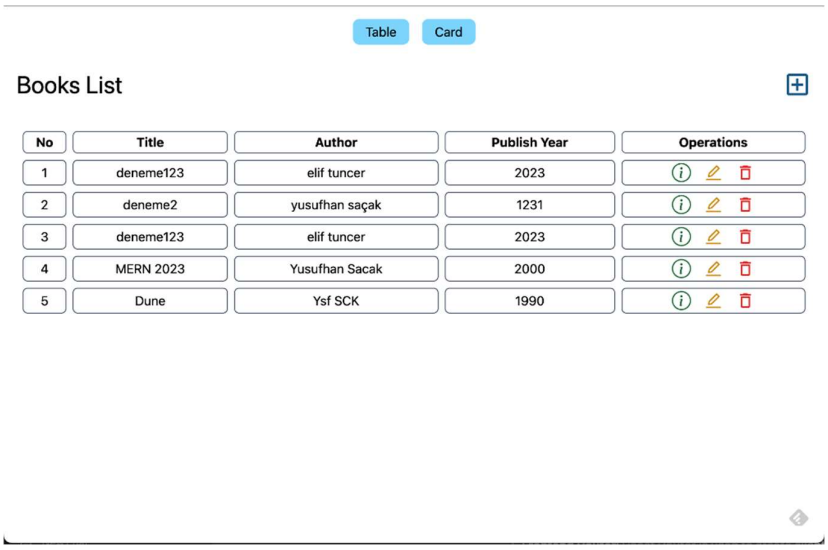
AIM: To Create a bookstore Library and Stock keeping APP:
a) User Interface: Browse Books from library, filter them based on category, author, publications, pay & rent them for a specific duration, like/review them
b) Admin interface: List/manage books, track rented books and their availability and send notifications via email to users once lease expire

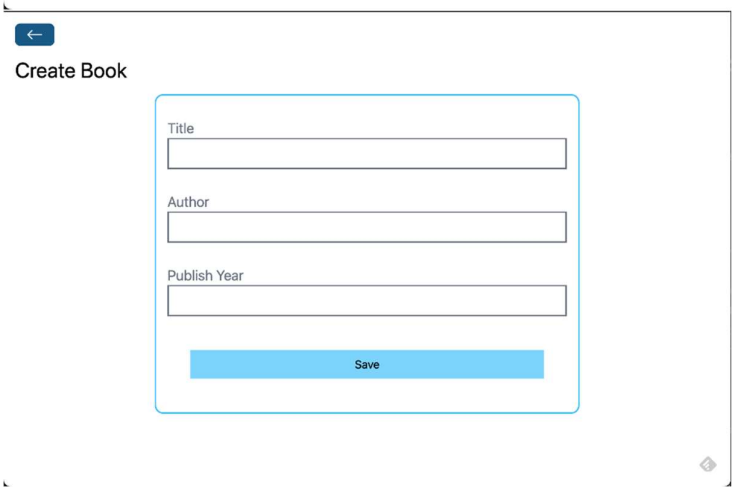
DESCRIPTION:

A library management application built using MongoDB, Express.js, React.js, and Node.js offers users and administrators a seamless experience. Users can browse and filter books, pay for rentals, and engage with the community by liking and reviewing books. Administrators can efficiently manage inventory, track rented books, and send automated email notifications for returns. This application enhances the book browsing and rental experience for all stakeholders.

TOOLS AND TECHNOLOGIES USED:

1) VS CODE
2) MongoDB, Expressjs, ReactJs, Nodejs





CODE:

Backend

```
import express, { request, response } from "express";
import { PORT, mongoDBURL } from "./config.js";
import mongoose from "mongoose";
import { Book } from "./models/bookModel.js";
import booksRoute from "./routes/booksRoute.js";
import cors from "cors";

const app = express();

// Middleware for parsing request body
app.use(express.json());

// Middleware for handling CORS POLICY
app.use(cors());
app.use(
  cors({
    origin: "https://localhost:3000",
    methods: ["GET", "POST", "PUT", "DELETE"],
    allowedHeaders: ["Content-Type"],
  })
);

app.get("/", (request, response) => {
  console.log(request);
  return response.status(234).send("Welcome to MERN Stack Book Shop");
});

app.use("/books", booksRoute);

mongoose
  .connect(mongoDBURL)
  .then(() => {
    console.log("App connected to db");
    app.listen(PORT, () => {
      console.log(`App listening on port ${PORT}!`);
    });
  })
  .catch((error) => {
    console.log(error);
  });        JosephDoUrden, 8 months ago • db connection done
```

Frontend

```
import React from 'react'
import {Routes, Route} from 'react-router-dom'

import Home from './pages/Home'
import CreateBook from './pages/CreateBooks'
import ShowBook from './pages/ShowBook'
import EditBook from './pages/EditBook'
import DeleteBook from './pages/DeleteBook'

const App = () => {
  return (
    <Routes>
      <Route path='/' element={<Home/>} />
      <Route path='/books/create' element={<CreateBook/>} />
      <Route path='/books/details/:id' element={<ShowBook/>} />
      <Route path='/books/edit/:id' element={<EditBook/>} />
      <Route path='/books/delete/:id' element={<DeleteBook/>} />
    </Routes>
  )
}

export default App        JosephDoUrden, 8 months ago • frontend packa
```

```
const Home = () => {
  const [books, setBooks] = useState([]);
  const [loading, setLoading] = useState(false);
  const [showType, setShowType] = useState("table");
  const { enqueueSnackbar } = useSnackbar();

  useEffect(() => {
    setLoading(true);
    axios
      .get("http://localhost:5555/books")
      .then((response) => {
        setBooks(response.data.data);
        setLoading(false);
      })
      .catch((error) => {
        console.log(error);
        enqueueSnackbar("Error", { variant: "error" });
        setLoading(false);
      });
  }, []);
  return (
    <div className="p-4">
      <div className="flex justify-center items-center gap-x-4">
        <button
          className="bg-sky-300 hover:bg-sky-600 px-4 py-1 rounded-lg"
          onClick={() => setShowType("table")}
        >
          Table
        </button>
        <button
          className="bg-sky-300 hover:bg-sky-600 px-4 py-1 rounded-lg"
          onClick={() => setShowType("card")}        JosephDoUrden, 8 months
        >
          Card
        </button>
      </div>
      <div className="flex justify-between items-center">
        <h1 className="text-3xl my-8">Books List</h1>
        <Link to="/books/create">
          <MdOutlineAddBox className="text-sky-800 text-4xl" />
        </Link>
      </div>
      {loading ? (
```

```
      {loading ? (
        <Spinner />
      ) : showType === "table" ? (
        <BooksTable books={books} />
      ) : (
        <BooksCard books={books} />
      )}
    </div>
  );
};

export default Home;
```

Experiment-7

AIM: Build a simple CRUD application: create a web application that allows users to create, Read, Update and Delete data from a MongoDB database.

DESCRIPTION:

Create, Read, Update, and Delete (CRUD) Application:

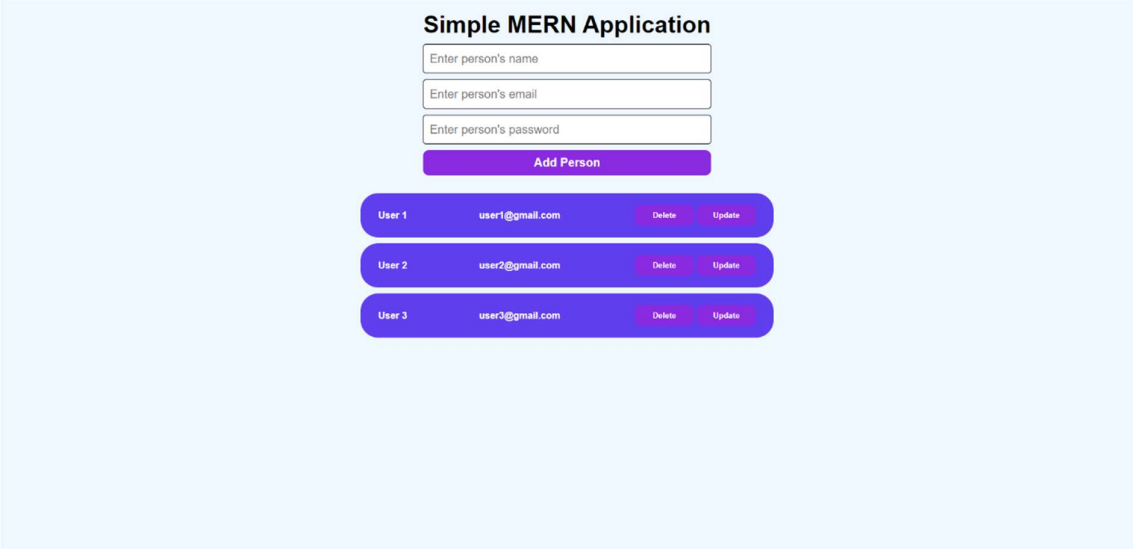
This web application allows users to perform basic database operations on a MongoDB database. Users can:

Create: Add new data entries to the database.
Read: View existing data stored in the database.
Update: Modify existing data entries in the database.
Delete: Remove data entries from the database.
Users interact with a user-friendly interface, where they can input new data, view existing data, edit data, and delete data as needed. This application provides a straightforward way to manage information stored in a MongoDB database.

TOOLS AND TECHNOLOGIES USED:

1) VS CODE
2) MongoDB, Expressjs, ReactJs, Nodejs

WEBPAGE



Code

Backend

```
require('dotenv').config();
require('express-async-errors');

const express = require("express");
const app = express();
const cors = require('cors');
const connectDB = require("./db/connect");
const peopleRouter = require("./routes/people");

app.use(express.json());
app.use(cors());
      Kuzma02, 4 months ago • "first_push"
app.use("/api/v1", peopleRouter);



const port = process.env.PORT || 3000;

const start = async () => {

    try {
        await connectDB(process.env.MONGO_URI);
        app.listen(port, () => {
            console.log("Server listening on port " + port);
        })
    } catch (error) {
        console.log(error);
    }

}

start();
```

Frontend

```
import { useState } from "react";
import { AddPerson, PersonList } from "./components";
import { ToastContainer, toast } from 'react-toastify';


function App() {

  const [ currentUpdatePerson, setCurrentUpdatePerson ] = useState(0);

  return (
    <div className="container-main">
      <AddPerson currentUpdatePerson={currentUpdatePerson} setCurrentUpdatePerson={setCurrentUpdatePerson} />
      <PersonList currentUpdatePerson={currentUpdatePerson} setCurrentUpdatePerson={setCurrentUpdatePerson} />
      <ToastContainer position='top-center' />
    </div>
  )
}

export default App
```

```
const PersonList = () => {        Kuzma02, 4 months ago • "async_thunk_implemented"

  const [ people, setPeople ] = useState([]);
  const { statusListener } = useSelector((state) => state.globalValues);

  const dispatch = useDispatch();


  const fetchData = async () => {

    try{
      // saljemo get(default) request
      const response = await axios(url);
      const data = response.data;
      setPeople(data.people);
    }catch(error){
      console.log(error);
    }
  }

  const deletePerson = async(id) => {
    try {
      await axios.delete(`http://localhost:3000/api/v1/people/${id}`);
      store.dispatch(changeStatusListener());
      toast.success("Person successfully deleted");
    } catch (error) {
      toast.error(error.message);
    }

  }

  const enterUpdateState = async (id) => {
    try{
      const response = await axios(`http://localhost:3000/api/v1/people/${id}`);
      const data = await response.data;
      dispatch(fetchPerson(data));
    }catch(error){
      toast.error(error.message);
    }

  }

  useEffect(() => {
    fetchData(url);
  useEffect(() => {
    fetchData(url);
  }, [statusListener])

  return (
    <div className="person-list-main">
      <ul>
        { people.map(item => (
        <li key={item._id}>
        <p>{ item.name }</p>
        <p>{ item.email }</p>
        <div className="person-list-btn-div">
          <button onClick={() => deletePerson(item._id)}>Delete</button>
          <button onClick={() => enterUpdateState(item._id)}>Update</button>
        </div>
      </li>
        )) }
      </ul>
    </div>
  );
};

export default PersonList;
```

Experiment- 8

AIM: Design a web platform to help small businesses manage their inventory using MERN stack.

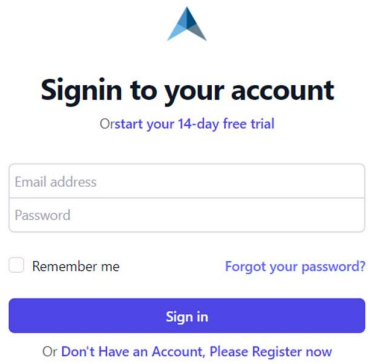
DESCRIPTION:

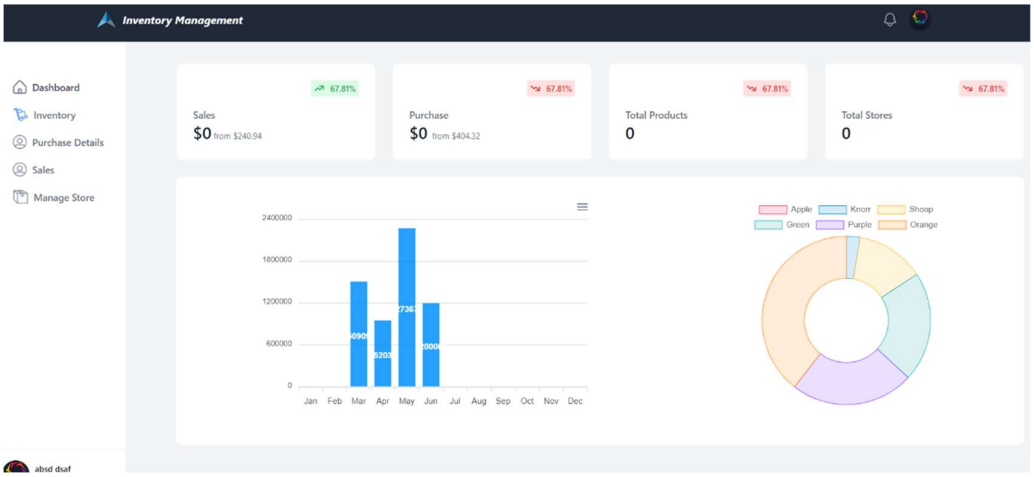
Our web platform, built with the MERN stack, assists small businesses in managing their inventory effectively. It provides a user-friendly interface for businesses to input, track, and update their inventory levels. Features include inventory categorization, stock level monitoring, and automated notifications for low stock. With intuitive controls and real-time updates, businesses can streamline their inventory management processes and make informed decisions to optimize stock levels and improve overall efficiency.

TOOLS AND TECHNOLOGIES USED:

1) VS CODE
2) MongoDB, Expressjs, ReactJs, Nodejs

WEBPAGE





Code
Backend

```
const express = require("express");
const { main } = require("./models/index");
const productRoute = require("./router/product");
const storeRoute = require("./router/store");
const purchaseRoute = require("./router/purchase");
const salesRoute = require("./router/sales");
const cors = require("cors");
const User = require("./models/users");
const Product = require("./models/Product");


const app = express();
const PORT = 4000;
main();
app.use(express.json());
app.use(cors());

// Store API
app.use("/api/store", storeRoute);

// Products API
app.use("/api/product", productRoute);

// Purchase API
app.use("/api/purchase", purchaseRoute);

// Sales API
app.use("/api/sales", salesRoute);

// ------------- Signin --------------
let userAuthCheck;
app.post("/api/login", async (req, res) => {
  console.log(req.body);
  // res.send("hi");
  try {
    const user = await User.findOne({
      email: req.body.email,
      password: req.body.password,
    });
    console.log("USER: ", user);
    if (user) {
      res.send(user);
      userAuthCheck = user;
    } else {
      res.status(401).send("Invalid Credentials");
      userAuthCheck = null;
    }
  } catch (error) {
    console.log(error);
    res.send(error);
  }
});

// Getting User Details of login user
app.get("/api/login", (req, res) => {
  res.send(userAuthCheck);
});
// -----------------------------------

// Registration API
app.post("/api/register", (req, res) => {
  let registerUser = new User({
    firstName: req.body.firstName,
    lastName: req.body.lastName,
    email: req.body.email,
    password: req.body.password,
    phoneNumber: req.body.phoneNumber,
    imageUrl: req.body.imageUrl,
  });

  registerUser
    .save()
    .then((result) => {
      res.status(200).send(result);
      alert("Signup Successfull");
    })
    .catch((err) => console.log("Signup: ", err));
  console.log("request: ", req.body);
});


app.get("/testget", async (req,res)=>{
  const result = await Product.findOne({ _id: '6429979b2e5434138eda1564'})
  res.json(result)

})

// Here we are listening to the server
app.listen(PORT, () => {
  console.log("I am live again");
});
```

Frontend

```
const App = () => {
  const [user, setUser] = useState("");
  const [loader, setLoader] = useState(true);
  let myLoginUser = JSON.parse(localStorage.getItem("user"));
  // console.log("USER: ",user)

  useEffect(() => {
    if (myLoginUser) {
      setUser(myLoginUser._id);
      setLoader(false);
      // console.log("inside effect", myLoginUser)
    } else {
      setUser("");
      setLoader(false);
    }
  }, [myLoginUser]);

  const signin = (newUser, callback) => {
    setUser(newUser);
    callback();
  };

  const signout = () => {
    setUser(null);
    localStorage.removeItem("user");
  };

  let value = { user, signin, signout };

  if (loader)
    return (
      <div
        style={{
          flex: 1,
          display: "flex",
          justifyContent: "center",
          alignItems: "center",
        }}
      >
        <h1>LOADING...</h1>
      </div>
    );

  return (
    <AuthContext.Provider value={value}>
      <BrowserRouter>
        <Routes>
          <Route path="/login" element={<Login />} />
          <Route path="/register" element={<Register />} />
          <Route
            path="/"
            element={
              <ProtectedWrapper>
                <Layout />
```

```
                <Layout />
              </ProtectedWrapper>
            }
          >
            <Route index element={<Dashboard />} />
            <Route path="/inventory" element={<Inventory />} />
            <Route path="/purchase-details" element={<PurchaseDetails />} />
            <Route path="/sales" element={<Sales />} />
            <Route path="/manage-store" element={<Store />} />
          </Route>
          <Route path="*" element={<NoPageFound />} />
        </Routes>
      </BrowserRouter>
    </AuthContext.Provider>
  );
};

export default App;
```