1. **Build a chat module using html, css and 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>Chat Module</title>

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

</head>

<body>

<div *class*="chat-container">

    <div *class*="chat-header">

        <h2>Chat Module</h2>

    </div>

    <div *class*="chat-messages" *id*="chatMessages">

*<!-- Messages will appear here -->*

    </div>

    <div *class*="chat-input">

        <input *type*="text" *id*="messageInput" *placeholder*="Type your message...">

        <button *onclick*="*sendMessage*()">Send</button>

    </div>

</div>

<script *src*="script.js"></script>

</body>

</html>

**script.js:**

*function* *sendMessage*() {

*var* messageInput *= document.getElementById(*"messageInput"*)*;

*var* chatMessages *= document.getElementById(*"chatMessages"*)*;

*if* (*messageInput*.*value*.*trim*() *!==* "") {

*var* message *= document.createElement(*"p"*)*;

*message*.*textContent* *=* *messageInput*.*value*;

*chatMessages*.*appendChild*(message);

*// Clear the input field*

*messageInput*.*value* *=* "";

*// Scroll to the bottom to show the latest message*

*chatMessages*.*scrollTop* *=* *chatMessages*.*scrollHeight*;

    }

}

**styles.css:**

body {

    font-family: Arial, sans-serif;

    margin: 0;

    padding: 0;

    box-sizing: border-box;

    background-color: rgb(40, 108, 143);

    display: flex;

    align-items: center;

    justify-content: center;

    height: 100*vh*;

}

*.chat-container* {

    border: 1*px* solid rgba(247, 247, 247, 0.659);

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

    width: 300*px*;

    overflow: hidden;

    display: flex;

    flex-direction: column;

}

*.chat-header* {

    background-color: #3498db;

    color: #fff;

    padding: 10*px*;

    text-align: center;

}

*.chat-messages* {

    flex-grow: 1;

    overflow-y: scroll;

    padding: 10*px*;

}

*.chat-input* {

    display: flex;

    justify-content: space-between;

    align-items: center;

    padding: 10*px*;

    background-color: rgb(51, 79, 130)

}

*.chat-input* input {

    flex-grow: 1;

    padding: 8*px*;

    margin-right: 10*px*;

}

*.chat-input* button {

    padding: 8*px* 12*px*;

    background-color: #3498db;

    color: #fff;

    border: none;

    cursor: pointer;

}

1. **Create a voting application using react JS**

**Commands:**  
npx create-react-app voting-app

cd voting-app

**App.js:**

*import* *React*, { *useState* } *from* 'react';

*import* './App.css';

*const VotingApp = () => {*

*const [options, setOptions] = useState([*

*{ id: 1, text:* 'Kaif'*, votes: 0 },*

*{ id: 2, text:* 'Madaesh'*, votes: 0 },*

*{ id: 3, text:* 'Dhiru'*, votes: 0 },*

*]);*

*const handleVote = (*optionId*) => {*

*setOptions((*prevOptions*) =>*

*prevOptions.map((*option*) =>*

*option.id ===* optionId *? { ...*option*, votes: option.votes + 1 } :* option

*)*

*);*

*};*

*return (*

*<*div *className=*"voting-app"*>*

*<*h1*>--------Who is ?--------</*h1*>*

*<*ul*>*

*{options.map((*option*) => (*

*<*li *key={option.id}>*

*{option.text} - Votes: {option.votes}*

*<*button *onClick={() => handleVote(option.id)}>Vote</*button*>*

*</*li*>*

*))}*

*</*ul*>*

*</*div*>*

*);*

*}*;

*export* *default* VotingApp;

**App.css:**

*.voting-app* {

  max-width: 600*px*;

  margin: auto;

  text-align: center;

  padding: 20*px*;

  background-color: #2b2323;

}

ul {

  list-style-type: none;

  padding: 0;

}

li {

  margin-bottom: 10*px*;

}

button {

  margin-left: 10*px*;

  cursor: pointer;

}

1. **Create a password strength checking application using node.js**

**Commands:**  
mkdir password-strength-checker

cd password-strength-checker

npm init -y

npm install express body-parser

npm install zxcvbn

node app.js

**app.js:**

*const express = require(*'express'*)*;

*const bodyParser = require(*'body-parser'*)*;

*const zxcvbn = require(*'zxcvbn'*)*;

*const app = express()*;

*const port = 3000*;

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

*app*.*use*(*express*.*static*('public'));

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

*res*.*sendFile*(\_\_dirname *+* '/public/index.html');

});

*app*.*post*('/check-password', (req, res) *=>* {

*const password = req.body.password*;

*const result = zxcvbn(*password*)*;

*res*.*json*({

        score: *result*.*score*,

        feedback: *result*.*feedback*.*suggestions*,

    });

});

*app*.*listen*(port, () *=>* {

*console*.*log*(`Server is running on http://localhost:${port}`);

});

**Index.html:**

<!DOCTYPE *html*>

<html *lang*="en">

<head>

    <meta *charset*="UTF-8">

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

    <title>Password Strength Checker</title>

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

</head>

<body>

    <div *class*="container">

        <h1>Password Strength Checker</h1>

        <label *for*="password">Enter your password:</label>

        <input *type*="password" *id*="password" *name*="password" *oninput*="*checkPassword*()">

        <div *id*="strength-meter"></div>

        <div *id*="feedback"></div>

    </div>

    <script>

*function* *checkPassword*() {

*const passwordInput = document.getElementById(*'password'*)*;

*const strengthMeter = document.getElementById(*'strength-meter'*)*;

*const feedback = document.getElementById(*'feedback'*)*;

*fetch*('/check-password', {

                method: 'POST',

                headers: {

                    'Content-Type': 'application/x-www-form-urlencoded',

                },

                body: `password=${*passwordInput*.*value*}`,

            })

            .*then*(response *=>* *response*.*json*())

            .*then*(data *=>* {

*strengthMeter*.*style*.*width* *=* `${(*data*.*score* *+* 1) *\** 20}%`;

*strengthMeter*.*className* *=* `strength-${*data*.*score*}`;

*feedback*.*innerHTML* *=* *data*.*feedback*.*join*('<br>');

            })

            .*catch*(error *=>* *console*.*error*(error));

        }

    </script>

</body>

</html>

**Styles.css:**

body {

    font-family: Arial, sans-serif;

    display: flex;

    align-items: center;

    justify-content: center;

    height: 100*vh*;

    margin: 0;

}

*.container* {

    text-align: center;

}

*#password* {

    margin-top: 10*px*;

    padding: 5*px*;

}

*#strength-meter* {

    height: 10*px*;

    background: #ddd;

    margin-top: 10*px*;

}

*.strength-0* {

    background: #ff6666;

}

*.strength-1* {

    background: #ffa07a;

}

*.strength-2* {

    background: #ffd700;

}

*.strength-3* {

    background: #add8e6;

}

*.strength-4* {

    background: #90ee90;

}

*#feedback* {

    margin-top: 10*px*;

    color: #666;

}

1. **Develop an application for grocery delivery using Angular JS**

**Commands:**

mkdir grocery-delivery-app

cd grocery-delivery-app

**index.html:**

<!DOCTYPE *html*>

<html *lang*="en" *ng-app*="groceryApp">

<head>

    <meta *charset*="UTF-8">

    <title>Grocery Delivery App</title>

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

    <script *src*="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>

    <script *src*="app.js"></script>

</head>

<body *ng-controller*="groceryController">

    <h1>Grocery Delivery App</h1>

    <div *ng-repeat*="item in groceryItems">

        <div *class*="grocery-item">

            <h2>{{ item.name }}</h2>

            <p>{{ item.description }}</p>

            <p>Price: ${{ item.price }}</p>

            <button *ng-click*="addToCart(item)">Add to Cart</button>

        </div>

    </div>

    <div *id*="cart">

        <h2>Shopping Cart</h2>

        <ul>

            <li *ng-repeat*="cartItem in shoppingCart">{{ cartItem.name }} - ${{ cartItem.price }}</li>

        </ul>

        <p>Total: ${{ calculateTotal() }}</p>

        <button *ng-click*="checkout()">Checkout</button>

    </div>

</body>

</html>

**App.js:**

*var* app *= angular.module(*'groceryApp'*, [])*;

*app*.*controller*('groceryController', *function* ($scope) {

*$scope*.*groceryItems* *=* [

        { name: 'Apples', description: 'Fresh red apples', price: 2.5 },

        { name: 'Bananas', description: 'Ripe yellow bananas', price: 1.8 },

        { name: 'Carrots', description: 'Organic carrots', price: 3.2 },

*// Add more grocery items as needed*

    ];

*$scope*.*shoppingCart* *=* [];

*$scope*.*addToCart* *=* *function* (item) {

*$scope*.*shoppingCart*.*push*({ name: *item*.*name*, price: *item*.*price* });

    };

*$scope*.*calculateTotal* *=* *function* () {

*var* total *= 0*;

*for* (*var* i *= 0*; i *<* *$scope*.*shoppingCart*.length; i*++*) {

            total *+=* *$scope*.*shoppingCart*[i].*price*;

        }

*return* *total*.*toFixed*(2);

    };

*$scope*.*checkout* *=* *function* () {

*alert*('Thank you for your order!');

*$scope*.*shoppingCart* *=* [];

    };

});

**Styles.css:**

body {

    font-family: Arial, sans-serif;

    text-align: center;

    margin: 20*px*;

}

h1, h2 {

    color: #333;

}

*.grocery-item* {

    border: 1*px* solid #ccc;

    padding: 10*px*;

    margin: 10*px*;

    display: inline-block;

    width: 200*px*;

}

button {

    background-color: #4CAF50;

    color: white;

    padding: 8*px* 15*px*;

    border: none;

    border-radius: 4*px*;

    cursor: pointer;

}

button*:hover* {

    background-color: #45a049;

}

*#cart* {

    margin-top: 20*px*;

}

ul {

    list-style-type: none;

    padding: 0;

}

li {

    margin: 5*px* 0;

}

1. **Develop an application for calculating BMI using Angular JS**

**Commans:**

mkdir my-angular-project

cd my-angular-project

**index.html:**

<!DOCTYPE *html*>

<html *lang*="en" *ng-app*="bmiCalculatorApp">

<head>

    <meta *charset*="UTF-8">

    <title>BMI Calculator</title>

    <script *src*="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>

    <style>

        body {

            font-family: Arial, sans-serif;

            text-align: center;

        }

*#calculator* {

            width: 300*px*;

            margin: 50*px* auto;

            padding: 20*px*;

            border: 1*px* solid #ccc;

            border-radius: 5*px*;

        }

    </style>

</head>

<body>

<div *id*="calculator" *ng-controller*="bmiCalculatorController">

    <h2>BMI Calculator</h2>

    <form>

        <label *for*="weight">Weight (kg): </label>

        <input *type*="number" *id*="weight" *ng-model*="weight" *required*>

        <br>

        <label *for*="height">Height (cm): </label>

        <input *type*="number" *id*="height" *ng-model*="height" *required*>

        <br>

        <button *ng-click*="calculateBMI()">Calculate BMI</button>

    </form>

    <br>

    <div *ng-show*="bmi">

        <h3>Your BMI is {{bmi.toFixed(2)}}</h3>

        <p>{{getBMICategory(bmi)}}</p>

    </div>

</div>

<script>

*var* app *= angular.module(*'bmiCalculatorApp'*, [])*;

*app*.*controller*('bmiCalculatorController', *function* ($scope) {

*$scope*.*calculateBMI* *=* *function* () {

*if* (*$scope*.*weight* *&&* *$scope*.*height*) {

*var* heightInMeters *= $scope.height / 100*;

*$scope*.*bmi* *=* *$scope*.*weight* */* (heightInMeters *\** heightInMeters);

            }

        };

*$scope*.*getBMICategory* *=* *function* (bmi) {

*if* (bmi *<* 18.5) {

*return* 'Underweight';

            } *else* *if* (bmi *>=* 18.5 *&&* bmi *<* 24.9) {

*return* 'Normal weight';

            } *else* *if* (bmi *>=* 25 *&&* bmi *<* 29.9) {

*return* 'Overweight';

            } *else* {

*return* 'Obese';

            }

        };

    });

</script>

</body>

</html>

1. **Create an offline image compressor using react JS and browser image compression**

**Commands:**

npx create-react-app image-compressor

cd image-compressor

npm install react-dropzone

npm start

**App.js:**

*// src/App.js*

*import* *React* *from* 'react';

*import* *ImageCompressor* *from* './ImageCompressor';

*function* *App*() {

*return* (

    <div *className=*"App">

      <h1>Offline Image Compressor</h1>

      <ImageCompressor />

    </div>

  );

}

*export* *default* App;

**ImageCompressor.js:**

*// src/components/ImageCompressor.js*

*import* *React*, { *useCallback*, *useState* } *from* 'react';

*import* { *useDropzone* } *from* 'react-dropzone';

*import* './components/ImageCompressor.css';

*const ImageCompressor = () => {*

*const [compressedImage, setCompressedImage] = useState(null);*

*const onDrop = useCallback((*acceptedFiles*) => {*

*const file =* acceptedFiles*[0];*

*if (*file*) {*

*compressImage(*file*);*

*}*

*}, []);*

*const compressImage = (*file*) => {*

*const reader = new FileReader();*

*reader.onload = (*event*) => {*

*const img = new Image();*

*img.src = event.target.result;*

*img.onload = () => {*

*const canvas = document.createElement(*'canvas'*);*

*const ctx = canvas.getContext(*'2d'*);*

*canvas.width = img.width;*

*canvas.height = img.height;*

*ctx.drawImage(*img*, 0, 0);*

*canvas.toBlob(*

*(*blob*) => {*

*setCompressedImage(*blob*);*

*},*

*file.type,*

*0.8*

*);*

*};*

*};*

*reader.readAsDataURL(*file*);*

*};*

*const { getRootProps, getInputProps } = useDropzone({* onDrop *});*

*return (*

*<*div *className=*"container"*>*

*<*div *{...getRootProps()} className=*"dropzone"*>*

*<*input *{...getInputProps()} />*

*<*p*>Drag & drop an image here, or click to select one</*p*>*

*</*div*>*

*{*compressedImage *&& (*

*<*div *className=*"image-preview"*>*

*<*h2*>Compressed Image:</*h2*>*

*<*img *src={URL.createObjectURL(*compressedImage*)} alt=*"Compressed" */>*

*</*div*>*

*)}*

*</*div*>*

*);*

*}*;

*export* *default* ImageCompressor;

**components/ImageCompressor.css:**

*/\* src/components/ImageCompressor.css \*/*

*.container* {

    text-align: center;

    margin: 50*px* auto;

  }

*.dropzone* {

    border: 2*px* dashed #ccc;

    border-radius: 4*px*;

    padding: 20*px*;

    cursor: pointer;

  }

*.image-preview* {

    margin-top: 20*px*;

  }

*.image-preview* img {

    max-width: 100*%*;

    max-height: 300*px*;

    border: 1*px* solid #ccc;

    border-radius: 4*px*;

  }

1. **Create a project for product catalog management using React JS**

**Commands:**

npx create-react-app voting-app

cd voting-app

**App.js:**

*import* *React*, { *useState* } *from* 'react';

*import* './App.css';

*import* *ProductList* *from* './ProductList';

*import* *AddProduct* *from* './AddProduct';

*import* *productsData* *from* './productsData';

*function* *App*() {

*const [products, setProducts] = useState(*productsData*)*;

*const handleAddProduct = (*newProduct*) => {*

*setProducts((*prevProducts*) => [...*prevProducts*,* newProduct*]);*

*}*;

*return* (

    <div *className=*"App">

      <h1>Product Catalog Management</h1>

      <ProductList *products=*{products} />

      <AddProduct *onAddProduct=*{handleAddProduct} />

    </div>

  );

}

*export* *default* App;

**ProductList.js:**

*import* *React* *from* 'react';

*import* *productsData* *from* './productsData';

*function* *ProductList*({ products }) {

*return* (

    <div>

      <h2>Product List</h2>

      <ul>

        {*products*.*map*((product) *=>* (

          <li *key=*{*product*.*id*}>

            {*product*.*name*} - ${*product*.*price*.*toFixed*(2)}

          </li>

        ))}

      </ul>

    </div>

  );

}

*export* *default* ProductList;

**AddProduct.js:**

*import* *React*, { *useState* } *from* 'react';

*function* *AddProduct*({ onAddProduct }) {

*const [productName, setProductName] = useState(*''*)*;

*const [productPrice, setProductPrice] = useState(*''*)*;

*const handleAddProduct = () => {*

*const newProduct = {*

*id: Date.now(),*

*name:* productName*,*

*price: parseFloat(*productPrice*),*

*};*

*onAddProduct(*newProduct*);*

*setProductName(*''*);*

*setProductPrice(*''*);*

*}*;

*return* (

    <div>

      <h2>Add Product</h2>

      <label>Name: </label>

      <input *type=*"text" *value=*{productName} *onChange=*{(e) *=>* *setProductName*(*e*.*target*.*value*)} />

      <br />

      <label>Price: </label>

      <input *type=*"text" *value=*{productPrice} *onChange=*{(e) *=>* *setProductPrice*(*e*.*target*.*value*)} />

      <br />

      <button *onClick=*{handleAddProduct}>Add Product</button>

    </div>

  );

}

*export* *default* AddProduct;

**productsData.js:**

*const productsData = [*

*{ id: 1, name:* 'Product 1'*, price: 10.99 },*

*{ id: 2, name:* 'Product 2'*, price: 19.99 },*

*{ id: 3, name:* 'Product 3'*, price: 29.99 },*

*]*;

*export* *default* productsData;

**App.css:**

*.App* {

  text-align: center;

  padding: 20*px*;

  background-color: aqua;

}

h1 {

  color: #333;

}

ul {

  list-style: none;

  padding: 0;

}

li {

  margin-bottom: 10*px*;

}

1. **Create a file sharing system using MongoDB**

**Commands:**

npm init

npm install express multer mongoose

npm install ejs

node server.js

**Project Structure:**

|- /uploads

|- /views

|- index.ejs

|- index.css

|- server.js

**Index.ejs:**

*<!-- views/index.ejs -->*

<!DOCTYPE *html*>

<html *lang*="en">

<head>

    <meta *charset*="UTF-8">

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

    <title>File Sharing System</title>

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

</head>

<body>

    <h1>File Sharing System</h1>

    <form *action*="/upload" *method*="post" *enctype*="multipart/form-data">

        <input *type*="file" *name*="file">

        <button *type*="submit">Upload</button>

    </form>

    <ul>

***<***% files.forEach(file => { %>

            <li>

***<***%= file.filename %>

                <a *href*="/download/***<***%= file.\_id %>">Download</a>

            </li>

***<***% }); %>

    </ul>

</body>

</html>

**Server.js:**

*const express = require(*'express'*)*;

*const multer = require(*'multer'*)*;

*const mongoose = require(*'mongoose'*)*;

*const path = require(*'path'*)*;

*const app = express()*;

*const port = 3000*;

*mongoose*.*connect*('mongodb://localhost/file\_sharing\_system', { useNewUrlParser: true, useUnifiedTopology: true });

*// Set EJS as the view engine*

*app*.*set*('view engine', 'ejs');

*app*.*set*('views', *path*.*join*(\_\_dirname, 'views'));

*const storage = multer.diskStorage({*

*destination:* './uploads/'*,*

*filename: function (*req*,* file*,* cb*) {*

*cb(null, file.originalname);*

*}*

*})*;

*const upload = multer({ storage:* storage *})*;

*const fileSchema = new mongoose.Schema({*

*filename:* String*,*

*path:* String*,*

*})*;

*const File = mongoose.model(*'File'*,* fileSchema*)*;

*app*.*get*('/', *async* (req, res) *=>* {

*const files = await File.find()*;

*res*.*render*('index', { files });

});

*app*.*post*('/upload', *upload*.*single*('file'), *async* (req, res) *=>* {

*const file = new File({*

*filename: req.file.originalname,*

*path: req.file.path,*

*})*;

*await* *file*.*save*();

*res*.*redirect*('/');

});

*app*.*get*('/download/:id', *async* (req, res) *=>* {

*const file = await File.findById(req.params.id)*;

*if* (file) {

*res*.*download*(*path*.*join*(\_\_dirname, *file*.*path*), *file*.*filename*);

    } *else* {

*res*.*status*(404).*send*('File not found');

    }

});

*app*.*listen*(port, () *=>* {

*console*.*log*(`Server is running on port ${port}`);

});

**Styles.css:**

body {

    font-family: 'Arial', sans-serif;

    background-color: #f4f4f4;

    margin: 0;

    padding: 0;

    display: flex;

    flex-direction: column;

    align-items: center;

    justify-content: center;

    height: 100*vh*;

}

h1 {

    color: #333;

}

form {

    display: flex;

    flex-direction: column;

    align-items: center;

    margin-top: 20*px*;

}

input[*type=*"file"] {

    margin-bottom: 10*px*;

}

button {

    padding: 10*px*;

    background-color: #4caf50;

    color: #fff;

    border: none;

    cursor: pointer;

}

button*:hover* {

    background-color: #45a049;

}

1. **Develop a habit tracking app with MongoDB, Node JS and Express**

**Commands:**

mkdir habit-tracking-app

cd habit-tracking-app

npm init -y

npm install express mongoose body-parser ejs

node app.js

**app.js:**

*const express = require(*'express'*)*;

*const mongoose = require(*'mongoose'*)*;

*const bodyParser = require(*'body-parser'*)*;

*const app = express()*;

*const port = 3000*;

*// Connect to MongoDB (Make sure your MongoDB server is running)*

*mongoose*.*connect*('mongodb://localhost:27017/habitTrackingApp', { useNewUrlParser: true, useUnifiedTopology: true });

*// Set up middleware*

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

*app*.*set*('view engine', 'ejs');

*app*.*use*(*express*.*static*('public'));

*// Define Habit model*

*const Habit = mongoose.model(*'Habit'*, {*

*name:* String*,*

*progress: { type:* Number*, default: 0 },*

*goal:* Number*,*

*})*;

*// Routes*

*app*.*get*('/', *async* (req, res) *=>* {

*try* {

*const habits = await Habit.find()*;

*res*.*render*('index', { habits });

    } *catch* (error) {

*console*.*error*(error);

*res*.*status*(500).*send*('Internal Server Error');

    }

});

*app*.*post*('/add', *async* (req, res) *=>* {

*try* {

*const { name, goal } = req.body*;

*const habit = new Habit({* name*,* goal *})*;

*await* *habit*.*save*();

*res*.*redirect*('/');

    } *catch* (error) {

*console*.*error*(error);

*res*.*status*(500).*send*('Internal Server Error');

    }

});

*app*.*post*('/update/:id', *async* (req, res) *=>* {

*try* {

*const habit = await Habit.findById(req.params.id)*;

*habit*.*progress* *+=* 1;

*await* *habit*.*save*();

*res*.*redirect*('/');

    } *catch* (error) {

*console*.*error*(error);

*res*.*status*(500).*send*('Internal Server Error');

    }

});

*// Start the server*

*app*.*listen*(port, () *=>* {

*console*.*log*(`Server is running on http://localhost:${port}`);

});

**Models/habit.js:**

*const mongoose = require(*'mongoose'*)*;

*const habitSchema = new mongoose.Schema({*

*name:* String*,*

*progress: { type:* Number*, default: 0 },*

*goal:* Number*,*

*})*;

module.exports *=* *mongoose*.*model*('Habit', habitSchema);

**views/index.ejs:**

<!DOCTYPE *html*>

<html *lang*="en">

<head>

    <meta *charset*="UTF-8">

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

    <title>Habit Tracking App</title>

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

</head>

<body>

    <h1>Habit Tracking App</h1>

    <ul>

***<***% habits.forEach(habit => { %>

            <li>

***<***%= habit.name %> - Progress: ***<***%= habit.progress %> / ***<***%= habit.goal %>

                <form *action*="/update/***<***%= habit.\_id %>" *method*="post" *style*="display: inline;">

                    <button *type*="submit">Update Progress</button>

                </form>

            </li>

***<***% }) %>

    </ul>

    <form *action*="/add" *method*="post">

        <label *for*="name">Habit Name:</label>

        <input *type*="text" *id*="name" *name*="name" *required*>

        <label *for*="goal">Goal:</label>

        <input *type*="number" *id*="goal" *name*="goal" *required*>

        <button *type*="submit">Add Habit</button>

    </form>

</body>

</html>

**Public/styles.css:**

body {

    font-family: Arial, sans-serif;

    text-align: center;

}

h1 {

    color: #333;

}

ul {

    list-style: none;

    padding: 0;

}

li {

    margin: 10*px* 0;

}

form {

    margin-top: 20*px*;

}

**13. Create a library management system using node.js**

**Commands:**

mkdir library-management-system

cd library-management-system

npm init -y

npm install express mongoose body-parser ejs

node app.js

**app.js:**

*const express = require(*'express'*)*;

*const mongoose = require(*'mongoose'*)*;

*const bodyParser = require(*'body-parser'*)*;

*const app = express()*;

*const port = 3000*;

*// Connect to MongoDB (Make sure your MongoDB server is running)*

*mongoose*.*connect*('mongodb://localhost:27017/libraryManagementSystem', { useNewUrlParser: true, useUnifiedTopology: true });

*// Set up middleware*

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

*app*.*set*('view engine', 'ejs');

*app*.*use*(*express*.*static*('public'));

*// Define Book model*

*const Book = mongoose.model(*'Book'*, {*

*title:* String*,*

*author:* String*,*

*ISBN:* String*,*

*available:* Boolean*,*

*})*;

*// Routes*

*app*.*get*('/', *async* (req, res) *=>* {

*try* {

*const books = await Book.find()*;

*res*.*render*('index', { books });

    } *catch* (error) {

*console*.*error*(error);

*res*.*status*(500).*send*('Internal Server Error');

    }

});

*app*.*get*('/add', (req, res) *=>* {

*res*.*render*('add');

});

*app*.*post*('/add', *async* (req, res) *=>* {

*try* {

*const { title, author, ISBN } = req.body*;

*const book = new Book({* title*,* author*,* ISBN*, available: true })*;

*await* *book*.*save*();

*res*.*redirect*('/');

    } *catch* (error) {

*console*.*error*(error);

*res*.*status*(500).*send*('Internal Server Error');

    }

});

*app*.*get*('/borrow/:id', *async* (req, res) *=>* {

*try* {

*const book = await Book.findById(req.params.id)*;

*if* (*book*.*available*) {

*book*.*available* *=* false;

*await* *book*.*save*();

        }

*res*.*redirect*('/');

    } *catch* (error) {

*console*.*error*(error);

*res*.*status*(500).*send*('Internal Server Error');

    }

});

*app*.*get*('/return/:id', *async* (req, res) *=>* {

*try* {

*const book = await Book.findById(req.params.id)*;

*if* (*!book*.*available*) {

*book*.*available* *=* true;

*await* *book*.*save*();

        }

*res*.*redirect*('/');

    } *catch* (error) {

*console*.*error*(error);

*res*.*status*(500).*send*('Internal Server Error');

    }

});

*// Start the server*

*app*.*listen*(port, () *=>* {

*console*.*log*(`Server is running on http://localhost:${port}`);

});

**Models/book.js:**

*const mongoose = require(*'mongoose'*)*;

*const bookSchema = new mongoose.Schema({*

*title:* String*,*

*author:* String*,*

*ISBN:* String*,*

*available:* Boolean*,*

*})*;

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

**views/add.ejs:**

<!DOCTYPE *html*>

<html *lang*="en">

<head>

    <meta *charset*="UTF-8">

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

    <title>Add a Book</title>

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

</head>

<body>

    <h1>Add a Book</h1>

    <form *action*="/add" *method*="post">

        <label *for*="title">Title:</label>

        <input *type*="text" *id*="title" *name*="title" *required*>

        <br>

        <label *for*="author">Author:</label>

        <input *type*="text" *id*="author" *name*="author" *required*>

        <br>

        <label *for*="ISBN">ISBN:</label>

        <input *type*="text" *id*="ISBN" *name*="ISBN" *required*>

        <br>

        <button *type*="submit">Add Book</button>

    </form>

    <a *href*="/">Back to Library</a>

</body>

</html>

**views/index.ejs:**

<!DOCTYPE *html*>

<html *lang*="en">

<head>

    <meta *charset*="UTF-8">

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

    <title>Library Management System</title>

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

</head>

<body>

    <h1>Library Management System</h1>

    <ul>

***<***% books.forEach(book => { %>

            <li>

***<***%= book.title %> by ***<***%= book.author %> (ISBN: ***<***%= book.ISBN %>)

***<***% if (book.available) { %>

                    <a *href*="/borrow/***<***%= book.\_id %>">Borrow</a>

***<***% } else { %>

                    <a *href*="/return/***<***%= book.\_id %>">Return</a>

***<***% } %>

            </li>

***<***% }) %>

    </ul>

    <a *href*="/add">Add a Book</a>

</body>

</html>

**Public/styles.css:**

body {

    font-family: Arial, sans-serif;

    text-align: center;

}

h1 {

    color: #333;

}

ul {

    list-style: none;

    padding: 0;

}

li {

    margin: 10*px* 0;

}

a {

    text-decoration: none;

    color: #007bff;

    margin-left: 10*px*;

}

**14. Develop any web application and include a user authentication system using node.js**

**Commands:**

mkdir node-auth-app

cd node-auth-app

npm init -y

npm install express mongoose express-session bcrypt body-parser ejs

node app.js

**app.js:**

*const express = require(*'express'*)*;

*const mongoose = require(*'mongoose'*)*;

*const session = require(*'express-session'*)*;

*const bcrypt = require(*'bcrypt'*)*;

*const bodyParser = require(*'body-parser'*)*;

*const app = express()*;

*const port = 3000*;

*// Connect to MongoDB (Make sure your MongoDB server is running)*

*mongoose*.*connect*('mongodb://localhost/nodeAuthApp', { useNewUrlParser: true, useUnifiedTopology: true });

*// Set up middleware*

*app*.*use*(*express*.*static*('public'));

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

*app*.*use*(*session*({

    secret: 'your-secret-key',

    resave: true,

    saveUninitialized: true

}));

*app*.*set*('view engine', 'ejs');

*// Define User model*

*const User = mongoose.model(*'User'*, {*

*username:* String*,*

*password:* String*,*

*})*;

*// Routes*

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

*res*.*render*('index', { user: *req*.*session*.*user* });

});

*app*.*get*('/register', (req, res) *=>* {

*res*.*render*('register');

});

*app*.*post*('/register', *async* (req, res) *=>* {

*try* {

*const { username, password } = req.body*;

*const hashedPassword = await bcrypt.hash(*password*, 10)*;

*const user = new User({* username*, password:* hashedPassword *})*;

*await* *user*.*save*();

*req*.*session*.*user* *=* user;

*res*.*redirect*('/');

    } *catch* (error) {

*console*.*error*(error);

*res*.*status*(500).*send*('Internal Server Error');

    }

});

*app*.*get*('/login', (req, res) *=>* {

*res*.*render*('login');

});

*app*.*post*('/login', *async* (req, res) *=>* {

*try* {

*const { username, password } = req.body*;

*const user = await User.findOne({* username *})*;

*if* (user *&&* *await* *bcrypt*.*compare*(password, *user*.*password*)) {

*req*.*session*.*user* *=* user;

*res*.*redirect*('/');

        } *else* {

*res*.*redirect*('/login');

        }

    } *catch* (error) {

*console*.*error*(error);

*res*.*status*(500).*send*('Internal Server Error');

    }

});

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

*req*.*session*.*destroy*();

*res*.*redirect*('/');

});

*// Start the server*

*app*.*listen*(port, () *=>* {

*console*.*log*(`Server is running on http://localhost:${port}`);

});

**Models/User.js:**

*const mongoose = require(*'mongoose'*)*;

*const userSchema = new mongoose.Schema({*

*username:* String*,*

*password:* String*,*

*})*;

module.exports *=* *mongoose*.*model*('User', userSchema);

**views/index.ejs:**

<!DOCTYPE *html*>

<html *lang*="en">

<head>

    <meta *charset*="UTF-8">

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

    <title>Node Auth App</title>

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

</head>

<body>

    <h1>Welcome ***<***%= user ? user.username : 'Guest' %>!</h1>

***<***% if (!user) { %>

        <p><a *href*="/register">Register</a> or <a *href*="/login">Login</a></p>

***<***% } else { %>

        <p><a *href*="/logout">Logout</a></p>

***<***% } %>

</body>

</html>

**Views/login.ejs:**

<!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*="/styles.css">

</head>

<body>

    <h1>Login</h1>

    <form *action*="/login" *method*="post">

        <label *for*="username">Username:</label>

        <input *type*="text" *id*="username" *name*="username" *required*>

        <br>

        <label *for*="password">Password:</label>

        <input *type*="password" *id*="password" *name*="password" *required*>

        <br>

        <button *type*="submit">Login</button>

    </form>

    <p>Don't have an account? <a *href*="/register">Register here</a></p>

</body>

</html>

**Views/register.ejs:**

<!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*="/styles.css">

</head>

<body>

    <h1>Register</h1>

    <form *action*="/register" *method*="post">

        <label *for*="username">Username:</label>

        <input *type*="text" *id*="username" *name*="username" *required*>

        <br>

        <label *for*="password">Password:</label>

        <input *type*="password" *id*="password" *name*="password" *required*>

        <br>

        <button *type*="submit">Register</button>

    </form>

    <p>Already have an account? <a *href*="/login">Login here</a></p>

</body>

</html>

**Public/styles.css:**

body {

    font-family: Arial, sans-serif;

    text-align: center;

    margin: 20*px*;

}

h1 {

    color: #333;

}

form {

    margin-top: 20*px*;

}

label, input {

    margin: 10*px*;

}

button {

    background-color: #4CAF50;

    color: white;

    padding: 8*px* 15*px*;

    border: none;

    border-radius: 4*px*;

    cursor: pointer;

}

button*:hover* {

    background-color: #45a049;

}

p {

    margin-top: 10*px*;

}

**19. Develop a simple dashboard for online shopping mart to perform ‘view products’**

**Index.html:**

<!DOCTYPE *html*>

<html *lang*="en">

<head>

    <meta *charset*="UTF-8">

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

    <title>Shopping Mart Dashboard</title>

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

</head>

<body>

    <div *class*="dashboard">

        <h1>Product Dashboard</h1>

        <div *class*="product-list" *id*="productList"></div>

    </div>

    <script *src*="products.js"></script>

</body>

</html>

**Styles.css:**

body {

    font-family: Arial, sans-serif;

    margin: 0;

}

*.dashboard* {

    padding: 20*px*;

}

h1 {

    color: #333;

}

*.product-card* {

    border: 1*px* solid #ddd;

    padding: 10*px*;

    margin: 10*px*;

    display: inline-block;

    width: 200*px*;

}

*.product-card* img {

    max-width: 100*%*;

    height: auto;

}

*.product-card* h2 {

    margin-top: 5*px*;

}

*.product-card* p {

    color: #666;

}

**Products.js:**

*const products = [*

*{*

*id: 1,*

*name:* 'Product 1'*,*

*description:* 'Lorem ipsum dolor sit amet, consectetur adipiscing elit.'*,*

*price: 19.99,*

*image:* 'https://via.placeholder.com/150'*,*

*},*

*{*

*id: 2,*

*name:* 'Product 2'*,*

*description:* 'Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris.'*,*

*price: 29.99,*

*image:* 'https://via.placeholder.com/150'*,*

*},*

*// Add more products as needed*

*]*;

*const productList = document.getElementById(*'productList'*)*;

*products*.*forEach*(product *=>* {

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

*productCard*.*classList*.*add*('product-card');

*productCard*.*innerHTML* *=* `

        <img src="${*product*.*image*}" alt="${*product*.*name*}">

        <h2>${*product*.*name*}</h2>

        <p>${*product*.*description*}</p>

        <p>Price: $${*product*.*price*.*toFixed*(2)}</p>

    `;

*productList*.*appendChild*(productCard);

});

**16. Create a simple micro blogging application to post text/multimedia content**

**Commands:**

mkdir micro-blogging-app

cd micro-blogging-app

npm init -y

npm install express mongoose express-session multer ejs

node app.js

**app.js:**

*const express = require(*'express'*)*;

*const mongoose = require(*'mongoose'*)*;

*const session = require(*'express-session'*)*;

*const multer = require(*'multer'*)*;

*const path = require(*'path'*)*;

*const app = express()*;

*const port = 3000*;

*// Connect to MongoDB (Make sure your MongoDB server is running)*

*mongoose*.*connect*('mongodb://localhost/microBloggingApp', { useNewUrlParser: true, useUnifiedTopology: true });

*// Set up middleware*

*app*.*use*(*express*.*static*('public'));

*app*.*use*(*express*.*json*());

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

*app*.*use*(*session*({

    secret: 'your-secret-key',

    resave: true,

    saveUninitialized: true

}));

*// Set EJS as the view engine*

*app*.*set*('view engine', 'ejs');

*// Set up Multer for handling file uploads*

*const storage = multer.diskStorage({*

*destination:* './public/uploads/'*,*

*filename: function (*req*,* file*,* cb*) {*

*cb(null, file.fieldname +* '-' *+ Date.now() + path.extname(file.originalname));*

*}*

*})*;

*const upload = multer({ storage:* storage *})*;

*// Define Post model*

*const Post = mongoose.model(*'Post'*, {*

*text:* String*,*

*image:* String*,*

*})*;

*// Routes*

*app*.*get*('/', *async* (req, res) *=>* {

*try* {

*const posts = await Post.find()*;

*res*.*render*('index', { posts });

    } *catch* (error) {

*console*.*error*(error);

*res*.*status*(500).*send*('Internal Server Error');

    }

});

*app*.*get*('/create', (req, res) *=>* {

*res*.*render*('create');

});

*app*.*post*('/create', *upload*.*single*('image'), *async* (req, res) *=>* {

*try* {

*const { text } = req.body*;

*const image = req.file ?* '/uploads/' *+ req.file.filename :* '';

*const post = new Post({* text*,* image *})*;

*await* *post*.*save*();

*res*.*redirect*('/');

    } *catch* (error) {

*console*.*error*(error);

*res*.*status*(500).*send*('Internal Server Error');

    }

});

*// Start the server*

*app*.*listen*(port, () *=>* {

*console*.*log*(`Server is running on http://localhost:${port}`);

});

**Models/Post.js:**

*const mongoose = require(*'mongoose'*)*;

*const postSchema = new mongoose.Schema({*

*text:* String*,*

*image:* String*,*

*})*;

module.exports *=* *mongoose*.*model*('Post', postSchema);

**views/index.ejs:**

<!DOCTYPE *html*>

<html *lang*="en">

<head>

    <meta *charset*="UTF-8">

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

    <title>Micro Blogging App</title>

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

</head>

<body>

    <h1>Micro Blogging App</h1>

    <div *id*="posts">

***<***% posts.forEach(post => { %>

            <div *class*="post">

                <p>***<***%= post.text %></p>

***<***% if (post.image) { %>

                    <img *src*="***<***%= post.image %>" *alt*="Post Image">

***<***% } %>

            </div>

***<***% }) %>

    </div>

    <a *href*="/create">Create a Post</a>

</body>

</html>

**Views/create.ejs:**

<!DOCTYPE *html*>

<html *lang*="en">

<head>

    <meta *charset*="UTF-8">

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

    <title>Create a Post</title>

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

</head>

<body>

    <h1>Create a Post</h1>

    <form *action*="/create" *method*="post" *enctype*="multipart/form-data">

        <label *for*="text">Text:</label>

        <textarea *id*="text" *name*="text" *rows*="4" *required*></textarea>

        <br>

        <label *for*="image">Image:</label>

        <input *type*="file" *id*="image" *name*="image">

        <br>

        <button *type*="submit">Create Post</button>

    </form>

    <a *href*="/">Back to Home</a>

</body>

</html>

**Public/styles.css:**

body {

    font-family: Arial, sans-serif;

    text-align: center;

    margin: 20*px*;

}

h1 {

    color: #333;

}

*#posts* {

    display: flex;

    flex-wrap: wrap;

    justify-content: center;

}

*.post* {

    border: 1*px* solid #ddd;

    padding: 10*px*;

    margin: 10*px*;

    display: inline-block;

    width: 300*px*;

    text-align: left;

}

img {

    max-width: 100*%*;

    height: auto;

}

a {

    display: block;

    margin-top: 10*px*;

}