**MVC**

/project

├── controllers

│ └── UserController.js

├── models

│ └── User.js

├── routes

│ └── userRoutes.js

├── views

│ ├── userForm.ejs

│ └── userCreated.ejs

├── app.js

└── package.json

// models/User.js

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const userSchema = new Schema({

    name: String,

    email: String

});

const User = mongoose.model('User', userSchema);

module.exports = User;

// controllers/UserController.js

const User = require('../models/User');

const showUserForm = (req, res) => {

    res.render('userForm');

};

const createUser = async (req, res) => {

    const { name, email } = req.body;

    try {

        const user = new User({ name, email });

        await user.save();

        res.render('userCreated', { name, email });

    } catch (err) {

        console.error(err);

        res.status(500).send('Error creating user');

    }

};

module.exports = { showUserForm, createUser };

// routes/userRoutes.js

const express = require('express');

const router = express.Router();

const { showUserForm, createUser } = require('../controllers/UserController');

router.get('/', showUserForm);

router.post('/createUser', createUser);

module.exports = router;

<!-- views/userForm.ejs -->

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>User Form</title>

</head>

<body>

    <h1>Create User</h1>

    <form action="/createUser" method="POST">

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

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

        <label for="email">Email:</label>

        <input type="email" id="email" name="email" required><br><br>

        <button type="submit">Submit</button>

    </form>

</body>

</html>

<!-- views/userCreated.ejs -->

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>User Created</title>

</head>

<body>

    <h1>User Created Successfully!</h1>

    <p>Name: <%= name %></p>

    <p>Email: <%= email %></p>

</body>

</html>

// app.js

const express = require('express');

const mongoose = require('mongoose');

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

const userRoutes = require('./routes/userRoutes');

const app = express();

const port = 3000;

// Set up EJS view engine

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

app.set('views', './views');

// Middleware

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

// MongoDB connection

mongoose.connect('mongodb://localhost:27017/myapp', {

    useNewUrlParser: true,

    useUnifiedTopology: true,

    useCreateIndex: true,

    useFindAndModify: false

});

const db = mongoose.connection;

db.on('error', console.error.bind(console, 'MongoDB connection error:'));

// Routes

app.use('/', userRoutes);

// Start the server

app.listen(port, () => {

    console.log(`App listening at http://localhost:${port}`);

});

**Export import nodejs**

**Exporting modules:**

// myModule.js

const myFunction = () => {

    console.log("Hello from myFunction!");

};

module.exports = {

    myFunction,

    myNumber: 123,

};

**Importing modules:**

// app.js

const myModule = require('./myModule');

myModule.myFunction(); // Outputs: Hello from myFunction!

console.log(myModule.myNumber); // Outputs: 123

**Exporting single values directly:**

// myValue.js

const myValue = 42;

module.exports = myValue;

**And then import it like this:**

// app.js

const myValue = require('./myValue');

console.log(myValue); // Outputs: 42

export a single value or function as the default export from a module:-

**Exporting a default value/function:**

// myModule.js

const myFunction = () => {

    console.log("Hello from myFunction!");

};

export default myFunction;

**Importing the default export:**

// app.js

import myFunction from './myModule';

myFunction(); // Outputs: Hello from myFunction!

**Exporting a default value directly:**

// myValue.js

export default 42;

And then import it like this:

// app.js

import myValue from './myValue';

console.log(myValue); // Outputs: 42

**module.exports**/**exports** are used in CommonJS modules (Node.js) to export multiple values, while **export default** is used in ECMAScript modules (ES6 modules) to export a single default value.

* Only one default export is allowed per module.
* When importing a default export, you can choose any name for it during import.

**Frontend->node.js**

(url parameters{'/user/:id'||req.params.id};post{req.body};get{req.query.q};HTTP Headers{ req.headers['custom-header']};File Upload{multer};WebSockets{socket.io})

**URL Parameters:**

const express = require('express');

const app = express();

app.get('/user/:id', (req, res) => {

    const userId = req.params.id;

    // Process userId and send response

    res.send(`User ID: ${userId}`);

});

app.listen(3000, () => {

    console.log('Server is running on port 3000');

});

<!-- Link with dynamic user ID -->

<a href="/user/1234">View User</a>

**Form Submission:**

const express = require('express');

const app = express();

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

    const { username, password } = req.body;

    // Process username and password

    res.send('Login successful');

});

app.listen(3000, () => {

    console.log('Server is running on port 3000');

});

<form action="/login" method="POST">

    <input type="text" name="username" placeholder="Username">

    <input type="password" name="password" placeholder="Password">

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

</form>

**AJAX Request (using fetch):**

const express = require('express');

const app = express();

app.post('/api/data', (req, res) => {

    const { data } = req.body;

    // Process received data

    res.json({ message: 'Data received successfully' });

});

app.listen(3000, () => {

    console.log('Server is running on port 3000');

});

<script>

    fetch('/api/data', {

        method: 'POST',

        headers: {

            'Content-Type': 'application/json'

        },

        body: JSON.stringify({ data: 'Hello from frontend' })

    })

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

    .then(data => console.log(data))

    .catch(error => console.error('Error:', error));

</script>

**Cookies or Sessions:**

const express = require('express');

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

const app = express();

app.use(session({

    secret: 'your-secret-key',

    resave: false,

    saveUninitialized: false

}));

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

    req.session.username = 'john.doe'; // Set session data

    res.send('Data set in session');

});

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

    const username = req.session.username; // Get session data

    res.send(`Username: ${username}`);

});

app.listen(3000, () => {

    console.log('Server is running on port 3000');

});

<!-- Example of using cookies in frontend -->

<script>

    document.cookie = "username=john.doe; expires=Thu, 18 Dec 2025 12:00:00 UTC; path=/";

// path=/->This attribute makes the cookie accessible on all pages within the domain. If not set, the cookie will only be accessible in the path of the current page where it was set.

</script>

**Query Parameters:**

const express = require('express');

const app = express();

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

    const searchTerm = req.query.q;

    // Process search term and send response

    res.send(`Search Query: ${searchTerm}`);

});

app.listen(3000, () => {

    console.log('Server is running on port 3000');

});

<!-- Form with action containing query parameters -->

<form action="/search" method="GET">

    <input type="text" name="q" placeholder="Search term">

    <button type="submit">Search</button>

</form>

**HTTP Headers:**

const express = require('express');

const app = express();

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

    const customHeader = req.headers['custom-header'];

    // Process custom header data and send response

    res.send(`Custom Header Value: ${customHeader}`);

});

app.listen(3000, () => {

    console.log('Server is running on port 3000');

});

<script>

    fetch('/headers', {

        headers: {

            'Custom-Header': 'my-custom-value'

        }

    })

    .then(response => response.text())

    .then(data => console.log(data))

    .catch(error => console.error('Error:', error));

</script>

**File Upload:**

const express = require('express');

const multer = require('multer');

const app = express();

const upload = multer({ dest: 'uploads/' });

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

    const uploadedFile = req.file;

    // Process uploaded file and send response

    res.send(`File Uploaded: ${uploadedFile.originalname}`);

});

app.listen(3000, () => {

    console.log('Server is running on port 3000');

});

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

    <input type="file" name="file">

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

</form>

const storage = multer.memoryStorage();

const upload = multer({ storage: storage });//to not storing and store in RAM.

//now store

const uploadedFile = req.file; // Access the uploaded file information const

filePath = path.join(\_\_dirname, 'uploads', uploadedFile.originalname); // Specify the path to save the file // Write the file to the filesystem

fs.writeFile(filePath, uploadedFile.buffer, (err) => { if (err) { return res.status(500).send('Error saving file.'); } res.send(`File uploaded and saved to: ${filePath}`); });

**WebSockets:**

const express = require('express');

const http = require('http');

const socketIo = require('socket.io');

const app = express();

const server = http.createServer(app);

const io = socketIo(server);

io.on('connection', socket => {

    console.log('A user connected');

    socket.on('chat message', msg => {

        console.log('Message received:', msg);

        // Broadcast the message to all connected clients

        io.emit('chat message', msg);

    });

    socket.on('disconnect', () => {

        console.log('User disconnected');

    });

});

server.listen(3000, () => {

    console.log('Socket.IO server running on port 3000');

});

<script src="/socket.io/socket.io.js"></script>

<script>

    const socket = io();

    // Example: sending a message from frontend to backend

    socket.emit('chat message', 'Hello from frontend');

    // Listening for messages from the backend

    socket.on('chat message', msg => {

        console.log('Message from server:', msg);

    });

</script>

**Node.js->Frontend**

(ejs{res.render('index', { data });ejs-><%= data.message %>};html{ res.sendFile(\_\_dirname + '/public/index.html', { data });html-><script>message ='<%= JSON.stringify(data.message)%>'</script>};FetchApi{res.json({mess:’hi’});})

Using EJS:

// app.js (Node.js backend)

const express = require('express');

const app = express();

const port = 3000;

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

app.use(express.static(\_\_dirname + '/public'));

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

  const data = {

    message: 'Hello from Node.js backend!'

  };

  res.render('index', { data });

});

app.listen(port, () => {

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

});

<!-- index.ejs (EJS template) -->

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>EJS Example</title>

</head>

<body>

  <h1><%= data.message %></h1>

</body>

</html>

1. Using plain HTML with Express:

// app.js (Node.js backend)

const express = require('express');

const app = express();

const port = 3000;

app.use(express.static(\_\_dirname + '/public'));

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

  const data = {

    message: 'Hello from Node.js backend!'

  };

  res.sendFile(\_\_dirname + '/public/index.html', { data });

});

app.listen(port, () => {

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

});

<!-- index.html (HTML file) -->

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>HTML Example</title>

</head>

<body>

  <h1 id="message"></h1>

  <script>

    const message = '<%= JSON.stringify(data.message) %>';

    document.getElementById('message').innerText = JSON.parse(message);

  </script>

</body>

</html>

**Using AJAX (Fetch API or XMLHttpRequest):**

// app.js (Node.js backend)

const express = require('express');

const app = express();

const port = 3000;

app.use(express.static(\_\_dirname + '/public'));

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

  const data = {

    message: 'Hello from Node.js backend!'

  };

  res.json(data);

});

app.listen(port, () => {

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

});

<!-- index.html (HTML file) -->

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>AJAX Example</title>

</head>

<body>

  <h1 id="message"></h1>

  <script>

    fetch('/data')

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

      .then(data => {

        document.getElementById('message').innerText = data.message;

      })

      .catch(error => console.error('Error fetching data:', error));

  </script>

</body>

</html>

**Using Templating Engines Directly in Express (Without EJS):**

// app.js (Node.js backend with Nunjucks)

const express = require('express');

const nunjucks = require('nunjucks');

const app = express();

const port = 3000;

nunjucks.configure('views', {

  autoescape: true,

  express: app

});

app.set('view engine', 'njk');

app.use(express.static(\_\_dirname + '/public'));

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

  const data = {

    message: 'Hello from Node.js backend!'

  };

  res.render('index', { data });

});

app.listen(port, () => {

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

});

<!-- views/index.njk (Nunjucks template) -->

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Nunjucks Example</title>

</head>

<body>

  <h1>{{ data.message }}</h1>

</body>

</html>

**Middle wears in expressjs**

**express.static**: This middleware is used to serve static files such as images, CSS files, and JavaScript files. It takes a root directory as an argument and serves files from that directory.

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

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Static HTML File</title>

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

</head>

<body>

  <h1>Welcome to My Static HTML Page</h1>

  <p>This is a sample HTML file served statically using Express.js.</p>

  <script src="script.js"></script>

</body>

</html>

**body-parser**: This middleware is used to parse the request body in different formats like JSON, URL-encoded, and raw. It makes the parsed data available in **req.body**.

const express = require('express');

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

const app = express();

// Parse JSON bodies

app.use(bodyParser.json());

// Parse URL-encoded bodies

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

// Route handler to handle POST request

app.post('/api/data', (req, res) => {

  console.log(req.body); // Access parsed request body

  res.send('Data received');

});

app.listen(3000, () => {

  console.log('Server is running on port 3000');

});

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>JSON POST Form</title>

</head>

<body>

  <h1>Send JSON Data</h1>

  <form id="jsonForm">

    <label for="name">Name:</label>

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

    <br>

    <label for="age">Age:</label>

    <input type="number" id="age" name="age">

    <br>

    <button type="submit">Submit</button>

  </form>

  <script>

    document.getElementById('jsonForm').addEventListener('submit', async (e) => {

      e.preventDefault();

      const formData = {

        name: document.getElementById('name').value,

        age: document.getElementById('age').value

      };

      const response = await fetch('/api/data', {

        method: 'POST',

        headers: {

          'Content-Type': 'application/json'

        },

        body: JSON.stringify(formData)

      });

      const data = await response.json();

      console.log(data);

    });

  </script>

</body>

</html>

**cookie-parser**: This middleware parses cookies attached to the client's request and makes them available in **req.cookies**.

const express = require('express');

const cookieParser = require('cookie-parser');

const app = express();

// Parse cookies

app.use(cookieParser());

// Route handler to handle cookie

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

  console.log(req.cookies); // Access parsed cookies

  res.send('Cookie received');

});

app.listen(3000, () => {

  console.log('Server is running on port 3000');

});

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Set Cookie</title>

</head>

<body>

  <h1>Set Cookie</h1>

  <form id="cookieForm">

    <label for="cookieName">Cookie Name:</label>

    <input type="text" id="cookieName" name="cookieName">

    <br>

    <label for="cookieValue">Cookie Value:</label>

    <input type="text" id="cookieValue" name="cookieValue">

    <br>

    <button type="submit">Set Cookie</button>

  </form>

  <script>

    document.getElementById('cookieForm').addEventListener('submit', (e) => {

      e.preventDefault();

      const cookieName = document.getElementById('cookieName').value;

      const cookieValue = document.getElementById('cookieValue').value;

      document.cookie = `${cookieName}=${cookieValue}`;

      alert('Cookie set successfully!');

    });

  </script>

</body>

</html>

**CRUD nodejs mongodb**

npm install express mongoose body-parser

// server.js

const express = require('express');

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

const mongoose = require('mongoose');

const app = express();

const PORT = 3000;

// MongoDB connection

mongoose.connect('mongodb://localhost:27017/mydb', {

  useNewUrlParser: true,

  useUnifiedTopology: true,

});

// Define a mongoose schema

const UserSchema = new mongoose.Schema({

  name: String,

  email: String,

});

const User = mongoose.model('User', UserSchema);

app.use(bodyParser.json());

// Create a new user

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

  try {

    const { name, email } = req.body;

    const newUser = new User({ name, email });

    await newUser.save();

    res.status(201).json(newUser);

  } catch (err) {

    res.status(400).json({ error: err.message });

  }

});

// Get all users

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

  try {

    const users = await User.find();

    res.status(200).json(users);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Update a user by ID

app.put('/users/:id', async (req, res) => {

  try {

    const { id } = req.params;

    const { name, email } = req.body;

    await User.findByIdAndUpdate(id, { name, email });

    res.status(200).json({ message: 'User updated successfully' });

  } catch (err) {

    res.status(400).json({ error: err.message });

  }

});

// Delete a user by ID

app.delete('/users/:id', async (req, res) => {

  try {

    const { id } = req.params;

    await User.findByIdAndDelete(id);

    res.status(200).json({ message: 'User deleted successfully' });

  } catch (err) {

    res.status(400).json({ error: err.message });

  }

});

// Start the server

app.listen(PORT, () => {

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

});

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>User Management</title>

</head>

<body>

  <h1>User Management</h1>

  <h2>Create User</h2>

  <form action="/users" method="POST">

    <label for="name">Name:</label>

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

    <label for="email">Email:</label>

    <input type="email" id="email" name="email" required><br>

    <button type="submit">Create User</button>

  </form>

  <hr>

  <h2>Update User</h2>

  <form action="/users/:id" method="PUT">

    <label for="userId">User ID:</label>

    <input type="text" id="userId" name="userId" required><br>

    <label for="updatedName">Updated Name:</label>

    <input type="text" id="updatedName" name="updatedName" required><br>

    <label for="updatedEmail">Updated Email:</label>

    <input type="email" id="updatedEmail" name="updatedEmail" required><br>

    <button type="submit">Update User</button>

  </form>

  <hr>

  <h2>Delete User</h2>

  <form action="/users/:id" method="DELETE">

    <label for="deleteUserId">User ID:</label>

    <input type="text" id="deleteUserId" name="deleteUserId" required><br>

    <button type="submit">Delete User</button>

  </form>

</body>

</html>