Started web application using react, node.js, mongodb, express, tailwindcss  
  
npm create vite@latest client

Framework: React

Variant: JS+SWC

Here we will have all dependencies: npm i

Search for tailwind vite and start following steps from documentation

DO GIT COMMIT

05/04/2025

Npm I react-router-dom

Check how many pages are need while creating website, here we need home,about,sign-in,sign-up, then create router with these pages

Make sure to check router is connecting with all pages or not

DO GIT COMMIT

Next is to create component folder as we checked in original website we have header for every pages, so we create header component

So we add them in APP.jsx after browserroute

When added tailwind, if there is not happening css in browser responsive, you can check in google developer tool (responsive browser)

DO GIT COMMIT

Create backend folder name as “api”

Install express, nodemon

Add type:”module” as it is ES6 module we need to add while using import express from “express”-----

  "name": "api",

  "version": "1.0.0",

  "description": "",

  "main": "index.js",

  "type":"module”

Delete test text in scripts in package.json and add dev, start   
then only we add npm run dev

DO GIT COMMIT

As we created git in client we need to move to api also

mv .git ../

created mongodb database -project and cluster, added in env (mongourl)

installed dotenv and mongoose

DO GIT COMMIT

We need to create test api route in backend by creating folders of router, controller and add json file in controller whether it is working or not. Check as we are using ES module, we need to import

DO GIT COMMIT

When create auth router check all slashes,

While req.body from controller we need to make sure to use

app.use(express.json()); add in server.js

we need to save data in db so we deconstruction the body (postman) and save in user schema which we have created then we need to use “user.save()”, as we know for every computer there will be change of time in delay to save in database, so we use await then async

Next we need to check for password as we save directly in database we need to encrypt it using bcryptsjs and add

const hashedPassword = bcrypt.hashSync(password, 8); after deconstruction

to get error visible in postman we use try and catch method

res.status(500).json(error.message)

but above error message is correct but the way of keeping is different so we will next commit

DO GIT COMMIT

We create middleware of file error.middleware.js

export const errhander = (err, req, res, next) => {

    const statusCode = err.statusCode || 500;

    const message = err.message || 'Internal Server Error';

    res.status(statusCode).json({

        status: 'error',

        statusCode,

        message,

    });

}

And add app.use(errhander) in server.js

Change the arrow funcntion controller by adding next and next(error) in catch method(try/catch)

Here error is coming from json file ,

We can create our own error code and message by using Utils folder

export const errorHandler = (statusCode, message) => {

    const error = new Error();

    error.statusCode = statusCode;

    error.message = message;

    return error;

}

Then in controller we can we give our statuscode and message

next(errorHandler(550, "User signup failed"));

**✅ What You Did in utils/error.js:**

// utils/error.js

export const errorHandler = (statusCode, message) => {

const error = new Error();

error.statusCode = statusCode;

error.message = message;

return error;

};

This is **not** the error-handling middleware itself — this is just a **utility** to create consistent error objects you can throw or pass into next().

**✅ Using It in Your Controller:**

import { errorHandler } from '../utils/error.js';

export const signupController = (req, res, next) => {

try {

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

if (!username || !email || !password) {

return next(errorHandler(400, "All fields are required"));

}

// rest of logic...

res.status(201).json({ message: "Signed up!", user: { username, email } });

} catch (err) {

next(err);

}

};

This creates a clean, centralized way to throw custom errors. 💯

**✅ Then Combine With Your Middleware:**

// middleware/error.middleware.js

export const globalErrorHandler = (err, req, res, next) => {

const status = err.statusCode || 500;

const message = err.message || "Something went wrong";

res.status(status).json({

success: false,

message,

});

};

And use in server.js:

import { globalErrorHandler } from './middleware/error.middleware.js';

app.use(globalErrorHandler);

**🧠 Summary: You're Combining 2 Things**

| **File** | **Purpose** | **Example** |
| --- | --- | --- |
| utils/error.js | Creates a custom error object | next(errorHandler(400, "Bad input")) |
| middleware/error.middleware.js | Handles any error thrown or passed with next(err) | app.use(globalErrorHandler) |

This is **best practice**. You’re doing it right. ✅

DO GIT COMMIT

We have added bg-color in index.css as color of body are same for all pages

Here margin mean “m” and padding means “p”

My means margin of y-axis and mt means margin at top

For username,email and password we need to change type for similarily. We need to consider disabled option also

Create Link option where we click it navigates to signin page

DO GIT COMMIT