* Create repo in github,
* Get it locally by using commands in VS code terminal:
* Git clone <https://github.com/AnitMariaJoseph/JJ-Constructions.git>
* cd jj-constructions
* This project has two parts:
* client — for your React/Next.js frontend
* server — for your Node.js backend
* mkdir client
* mkdir client
* cd client

**✅ Step 1.4: Initialize the Frontend (Next.js)**

We’ll use **Next.js** (React-based) for the frontend.

* Run the Next.js setup command:
* npx create-next-app@latest .

**🔹 npx - Stands for:** Node Package Execute**.** Lets you run Node.js packages **without installing them globally**.It looks for a package (like create-next-app), downloads it temporarily if needed, and runs it. create-next-app is the official Next.js project scaffolding tool.

Example: Instead of doing npm install -g create-next-app, you can just use npx create-next-app.

| **Command** | **Installs Permanently?** | **Always Up to Date?** | **Best For** |
| --- | --- | --- | --- |
| npm install -g create-next-app | ✅ Yes | ❌ No | Frequent use, offline needs |
| npx create-next-app | ❌ No | ✅ Yes | One-off projects, beginners, staying current |

If error saying npx is not found, check node version (node -v), if not installed, install the LTS version from node website. Then try the above code. VS code uses powershell, hence npx might not execute doe to policy, to override the session use(Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass) and then use the above command. OR In VS Code, Ctrl + `, Click the dropdown arrow (▼) next to the **+** icon in the terminal, Select **"Command Prompt"** (or "cmd").If you don’t see Command Prompt listed, click **"Select Default Profile"** → choose **Command Prompt**, then reopen the terminal.

**✅ Step 1.5: Initialize the Backend (Node.js + Express)**

* cd server

1. Initialize a Node.js project:

* npm init -y

**npm init**

* This command **initializes a new Node.js project**. It creates a package.json file, which keeps track of: Project name, version, description, Dependencies, Scripts (like start, build, etc.)

**🔹 -y :** Short for –yes, **Skips all the prompts** and uses **default values** automatically

* cd ..

**✅ Step 1.6: Save and Push to GitHub**

Let’s commit this folder structure and push it to GitHub. Run these commands in the **root folder (jj-constructions)**:

* git add .

This stages all the changes.

* git commit -m "Initial project structure: Next.js frontend and Node.js backend"

This saves your changes locally.

* git push origin main

This uploads the changes to GitHub.

Created wireframe for the home page

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

Starting with the website coding, next.js project with tailwind css

* Move to client folder where Node.js app will be created

 Next.js is a React framework that makes building modern web apps easier.

 Tailwind CSS is a utility-first CSS framework for fast styling.

**Step 1: Run the development server to test**

Let’s first see if everything is already working!

In the terminal inside the client folder, run:

* npm install
* npm run dev

Then open your browser and go to:

* http://localhost:3000

You should see the default Next.js starter page. If you do — **you’re already set up!**

If you see a folder named \*\*app/\*\*, you're using the **App Router**.  
  
**Your frontend folder likely looks like this:**

client/

├── app/

│ ├── layout.tsx ← shared layout (like header/footer)

│ └── page.tsx ← homepage route

├── components/ ← reusable UI elements (we’ll add this)

├── public/ ← static assets like images

├── styles/ ← Tailwind styles if needed

├── tailwind.config.ts

├── tsconfig.json

├── package.json

📁 app/page.tsx → This is your homepage (/)  
📁 app/layout.tsx → Wraps all pages in a common layout (like <Header />)

**📁 1. Create the Folder**

In your project structure:

cpp

CopyEdit

client/

├── app/

├── public/

├── styles/ (or globals.css)

├── components/ ✅ ← Create this manually if it doesn't exist

You can do this from:

* **VS Code File Explorer**:
  + Right-click on client/ → click **New Folder** → name it components

OR

* **Terminal** (inside client/):

mkdir components

**📄 2. Create Header.tsx File**

Inside the new components folder, create a file called Header.tsx.

In VS Code:

* Right-click on components/ → **New File** → name it Header.tsx

Then paste this starter code into it:

// client/components/Header.tsx

export default function Header() {

return (

<header className="w-full px-6 py-4 shadow bg-white">

<div className="max-w-7xl mx-auto flex justify-between items-center">

<h1 className="text-xl font-bold">J & J Constructions</h1>

<nav className="space-x-4">

<a href="#" className="hover:underline">Home</a>

<a href="#" className="hover:underline">About</a>

<a href="#" className="hover:underline">Services</a>

<a href="#" className="hover:underline">Projects</a>

<a href="#" className="hover:underline">Contact</a>

</nav>

</div>

</header>

);

}

**📥 3. Import the Header in app/layout.tsx**

Now that the file exists, your import in layout.tsx should work correctly:

import Header from '@/components/Header'; // ✅ if you have alias set up

// OR, if not using alias yet:

import Header from '../components/Header'; // ✅ works with relative path

**🔄 4. Restart the Dev Server**

If you had the dev server running, restart it so it picks up the new file:

npm run dev

---------@ not recognised

**✅ Step-by-Step Fix: Set Up the @ Alias in Next.js + TypeScript**

**1. Open tsconfig.json in the root of your project.**

Look for or add this block:

{

"compilerOptions": {

"baseUrl": ".",

"paths": {

"@/\*": ["./\*"]

}

}

}

✅ This tells TypeScript that @ should point to the root of your project.

**2. Restart Your VS Code (or its TypeScript server)**

* After modifying tsconfig.json, VS Code sometimes needs to be restarted to pick up the changes.

Hero section:

client/components/Hero.tsx

Then, paste this example code:

tsx

Copy

Edit

// client/components/Hero.tsx

export default function Hero() {

return (

<section className="w-full bg-gray-50 py-20">

<div className="max-w-7xl mx-auto px-6 text-center">

<h2 className="text-4xl md:text-5xl font-bold text-gray-900">

Building Dreams into Reality

</h2>

<p className="mt-4 text-lg text-gray-600 max-w-2xl mx-auto">

At J & J Constructions, we specialize in quality craftsmanship for residential, commercial, and renovation projects.

</p>

<button className="mt-8 bg-blue-600 text-white px-6 py-3 rounded-lg hover:bg-blue-700 transition">

Get a Free Quote

</button>

</div>

</section>

);

}

🧩 Step 2: Import Hero in app/page.tsx

Open your homepage file:

tsx

Copy

Edit

// client/app/page.tsx

import Hero from '@/components/Hero'; // or use relative path

Then update your default export to use it:

tsx

Copy

Edit

export default function Home() {

return (

<main>

<Hero />

</main>

);

}