Introduction to Master Project

It's time for the most exciting part of the course, we haven't had the chance to explore every feature

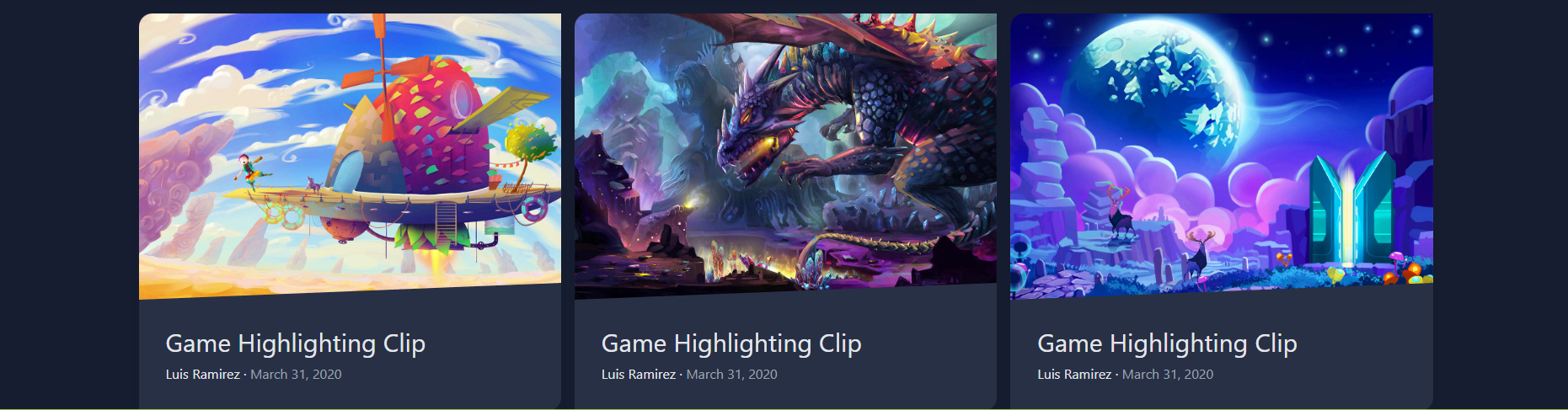
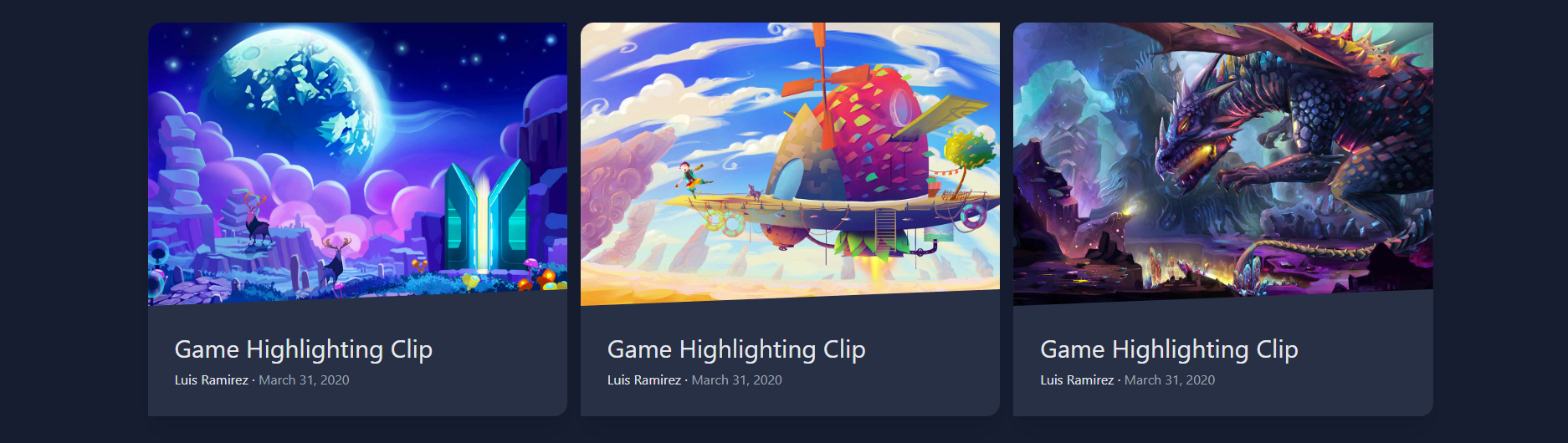
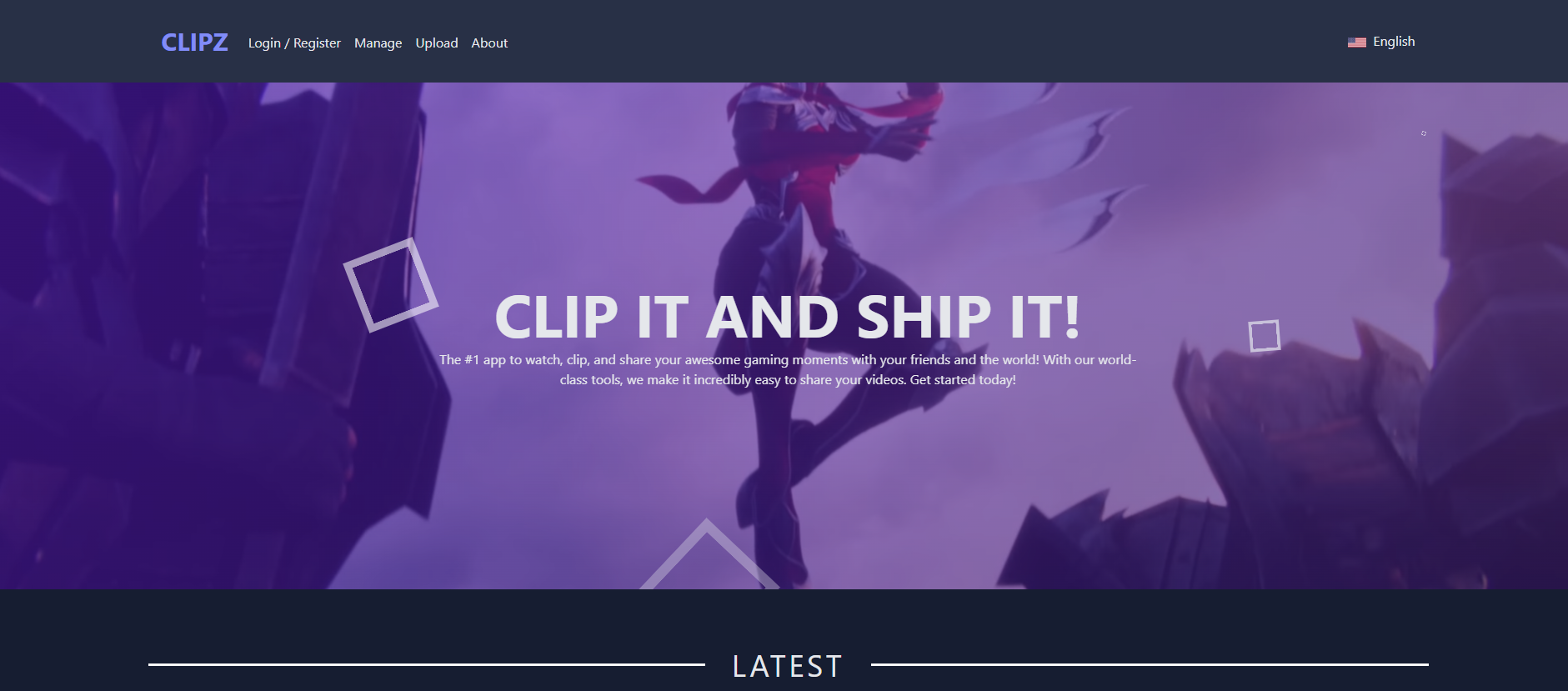
in Angular. However, we've covered most of the fundamentals. If we're going to continue further, we should work on a realistic project by working on a real project, we’ll get a chance to learn the best practices for structuring a project. So, what are we going to build?

We will develop a project called **Clipz,** a platform for uploading and sharing gaming highlights.

Features include authentication, routing, video processing, state management and so much more.

One of the most interesting features is running a web assembly file to process video files with FFmpeg. We will be playing with cutting edge features, which will surely impress recruiters. On

my screen, I'm showing the static design of the app we will be developing:



It doesn't look like much, but believe me, it's going to take a lot of work to convert this template

to a functioning app.

Once we're finished with our project, we will be deploying it. Throughout the development lifecycle

of this project, you will be surprised by how helpful Angular can be.

First, we need to create a new project.

Open the command line in your editor. We'll be working in a completely new directory. We aren't going to be installing our project with the basics app we worked on previously.

Inside the command line run the following command:

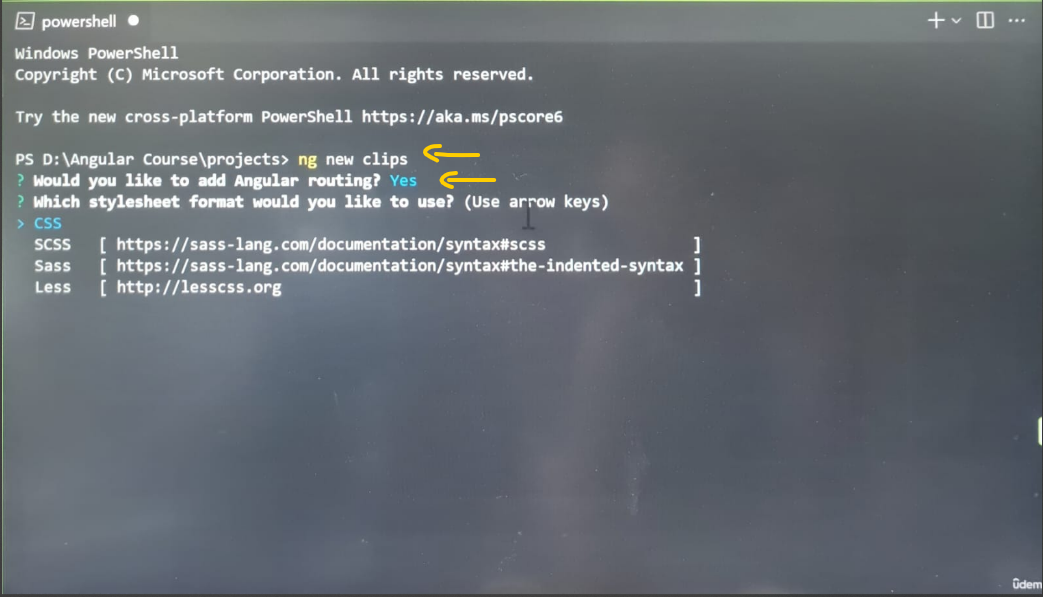
**ng new clips**

The CLI will ask us if we'd like to add **routing** for this project.

We will select Yes.

**Routing** is an optional feature. Larger apps typically have **routing**.

We will discuss how **routing** works in this course.



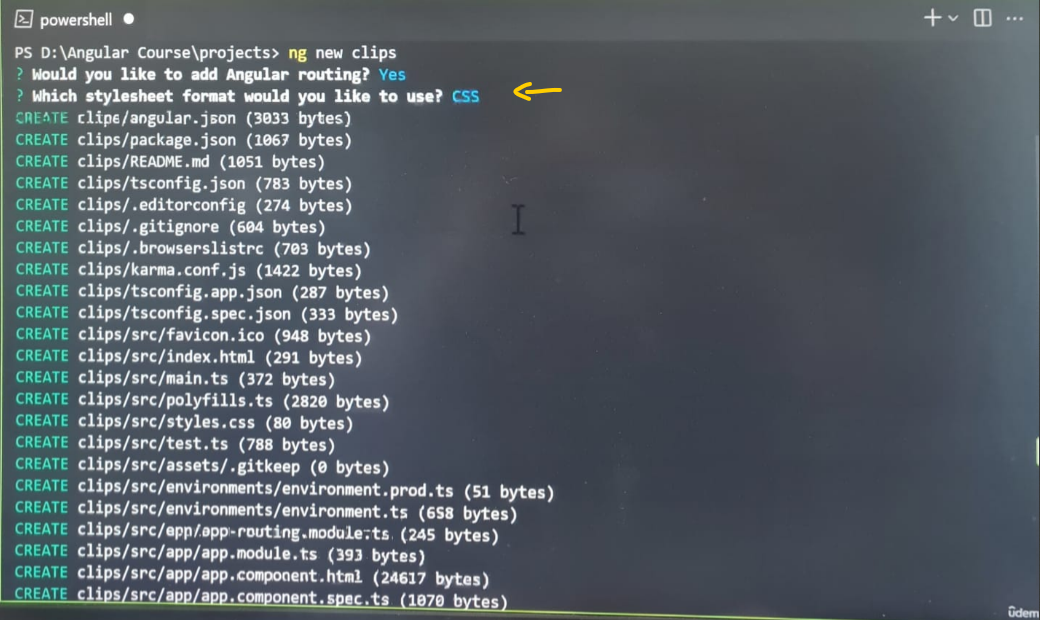
The next option will ask us to select a **CSS Preprocessor**.

We're not going to be using a preprocessor.

However, feel free to choose or preprocessor if you are familiar with any of them from this list.

After selecting our options, the CLI will be again creating and installing the project.

This installation may take a while.



**On my PC with Angular 20.0.5, I am getting different options for installation:**

**A conversation between me and ChatGPT:**

I have Angular 20 installed on my pc, I want to create a new project called clips with routing and css configured, what options should I use ?

**ChatGPT said:**

To create a new Angular 20 project named clips with **routing enabled** and **CSS for styling**, you can use the following CLI command:

bash

CopyEdit

ng new clips --routing --style=css

**🔍 Breakdown of Options:**

| **Option** | **What it does** |
| --- | --- |
| clips | Project name |
| --routing | Adds Angular Router configuration (creates app.routes.ts in standalone) |
| --style=css | Sets CSS as the default styling language (instead of SCSS, Sass, etc.) |

**✅ What Happens After Running This:**

* It creates a folder called clips
* Sets up a standalone Angular app (in Angular 20 this is default)
* Adds routing using a app.routes.ts file (instead of app-routing.module.ts)
* Configures the project to use plain CSS

**👉 What to Do Next:**

1. Navigate into the project:

bash

CopyEdit

cd clips

1. Serve the app locally:

bash

CopyEdit

ng serve

1. Open browser at: http://localhost:4200

Would you like me to explain the routing file setup in Angular 20 (standalone), or how to add new pages/components using routing?

**You said:**

Do you want to create a 'zoneless' application without zone.js (Developer Preview)? (y/N)

**ChatGPT said:**

That prompt means Angular CLI is giving you the **option to create a "zoneless" application**, which is a **new experimental feature** introduced in Angular 17+.

**🔍 What does it mean?**

* **Zone.js** is used by default in Angular to detect changes in the UI and trigger re-renders.
* A **zoneless app** skips zone.js and uses **signals** or manual change detection instead.
* Zoneless apps are **faster**, more **predictable**, and **leaner**, but also **require more manual control** over change detection.

**✅ For most users, especially when starting out:**

**Choose No** by pressing Enter or typing n.

txt

CopyEdit

Do you want to create a 'zoneless' application without zone.js (Developer Preview)? (y/N)

Just press Enter to accept the default (N = No), which is recommended unless:

* You're building a performance-critical app,
* You're familiar with Angular Signals and manual change detection,
* Or you want to experiment with the latest cutting-edge features.

Let me know if you want a side-by-side comparison of zoned vs zoneless apps or a guide for when to choose which.

**You said:**

? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? (y/N)

**ChatGPT said:**

That prompt is asking if you want to enable **Angular Universal**, which adds:

* ✅ **Server-Side Rendering (SSR)** — Angular renders the initial HTML on the server, improving SEO and performance.
* ✅ **Static Site Generation (SSG)** — Angular pre-renders pages at build time, useful for blogs or mostly static content.

**🔍 Should you enable it?**

| **Your Situation** | **Recommendation** |
| --- | --- |
| **Beginner** or building a **basic app** like "clips" | ❌ **No** (just press Enter) |
| Building a **SEO-heavy** site (e.g., marketing page, blog) | ✅ Yes |
| Building a **dynamic app or dashboard** | ❌ No |
| Want to learn SSR/SSG later | ❌ No for now, can add it later using ng add @angular/ssr |

**🧠 Summary**

For your use case:

🟢 Just press Enter or type n — you don’t need SSR/SSG for now.

You’ll get a faster and simpler development experience without the extra complexity.

Would you like me to show how to add SSR later if you ever need it?

**You said:**

The app I am building is a platform for uploading and sharing gaming highlights

**ChatGPT said:**

Great! Since you're building a **gaming highlights platform** — something like a **video-sharing web app** — here’s what I recommend for those Angular CLI prompts:

**✅ Final CLI Setup for Your Case**

Run:

bash

CopyEdit

ng new clips --routing --style=css

And when prompted:

**➤ Do you want to create a 'zoneless' application without zone.js?**

**Answer:** No (or just press Enter)

**➤ Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)?**

**Answer:** No (or just press Enter)

**🧠 Why?**

**❌ Zoneless:**

Your app will likely rely on Angular's standard change detection to keep components in sync with video uploads, likes, comments, etc. Going zoneless is advanced and unnecessary right now.

**❌ SSR/SSG:**

SSR helps with SEO, but gaming highlight platforms usually require:

* Client-heavy features (video playback, user interactions)
* Realtime updates
* Authenticated content

SSR adds complexity and isn’t essential unless you're optimizing for search engine indexing of public content.

**✅ What to Focus On Instead**

Since it’s a video-sharing platform, here's what you'll need:

| **Feature** | **Angular Tool/Tech** |
| --- | --- |
| File/video upload | Angular with Firebase Storage or Node.js backend |
| Routing between pages | --routing already sets this up |
| Authentication | Firebase Auth / Auth0 |
| Realtime updates (likes/comments) | Firebase Realtime DB / Firestore |
| Video embedding & playback | HTML5 <video> tag or libraries like Plyr |
| Responsive UI | Angular Material / Tailwind CSS |

After this setup as proposed, will I have the following file app-routing.module.ts inside src/app directory ?

**ChatGPT said:**

❌ **No**, you will **not** get app-routing.module.ts in Angular 20 if you're using the default (standalone) setup — even when you use the --routing flag.

**✅ What You’ll Get Instead (Angular 20 with Standalone API)**

After running:

bash

CopyEdit

ng new clips --routing --style=css

You’ll get a **file called**:

bash

CopyEdit

src/app/app.routes.ts

This file contains a **routes array** that looks like this:

ts

CopyEdit

import { Routes } from '@angular/router';

export const routes: Routes = [];

**🧠 Why the Change?**

Angular 15+ introduced the **standalone API**, and as of Angular 17–20, it’s the **default**. This simplifies the app by removing the need for NgModules like AppRoutingModule.

In standalone Angular:

* Components can directly declare their own routing imports
* app.routes.ts replaces app-routing.module.ts

**If You *Do* Want app-routing.module.ts**

If you prefer the old, module-based style (e.g., for tutorials or legacy compatibility), you can **opt out of standalone** by adding:

bash

CopyEdit

ng new clips --routing --style=css --standalone=false

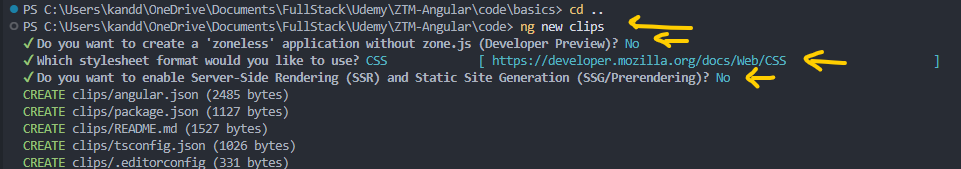
That will generate:

arduino

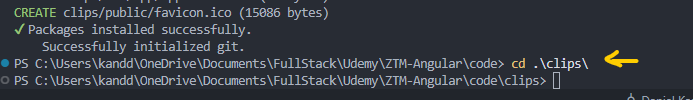
CopyEdit

src/app/app-routing.module.ts

src/app/app.module.ts

****

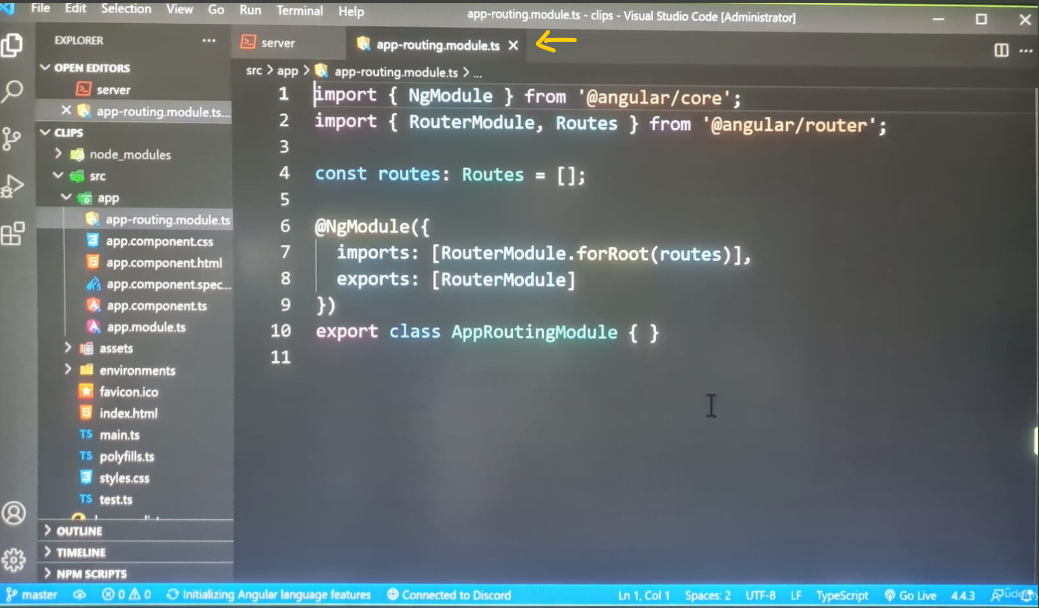
After the installation is complete, let's move in to the new project.

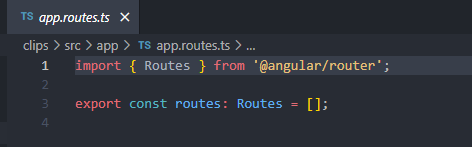


There aren't significant changes to the default project.

The most noticeable difference is a new file called **app-routing.module.ts** inside the **src/app**

directory. **[In my case, I don’t have this file. Instead, I have src/app/app.routes.ts – see the conversation with ChatGPT above].**

****

****

We will be exploring this file in an upcoming lecture, Angular has provided us with some **routes** for

the default project. We can safely ignore it for the time being.

Other than that, we're good to go.

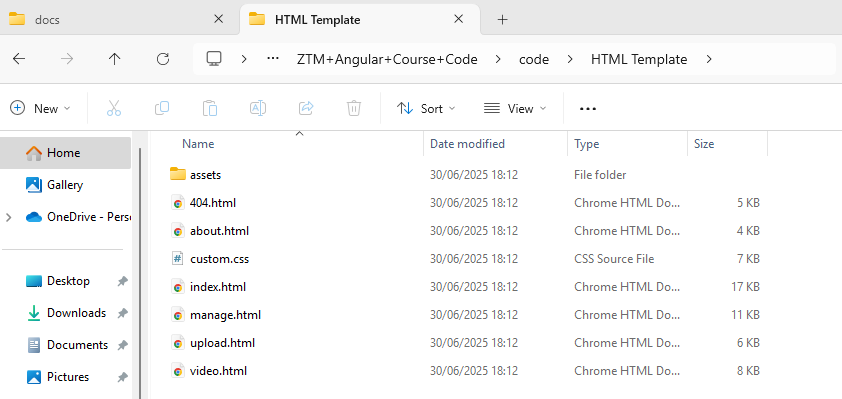
Now the question is where do we begin? What's the first step we should take for developing an app?

It can be a daunting task to get started.

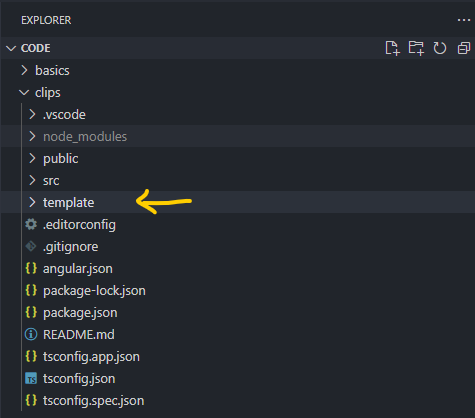
From my experience, I like to start by importing the static design. We already have the template in hands. We should start adding it to our project.

In the resource section of this lecture. I provide a zip file with the HTML and CSS for our app.

It's completely static. I haven't done anything to this template to prepare it for Angular.



We are going to make it functional from top to bottom. Download this file before moving on. After downloading the zip file. Create a new directory in our project called **templates**.



This is not a normal folder to have in a project. It's not going to stick around, once we've converted the templates, it can be safely removed.

We don't have to worry about Angular doing anything with this folder.

The purpose of this folder is to temporarily store our template files so that we can reference them.

Drag and drop the template files from the zip file to this new folder.

After doing so, we can start to transfer the templates.

We will begin with the **home page**.

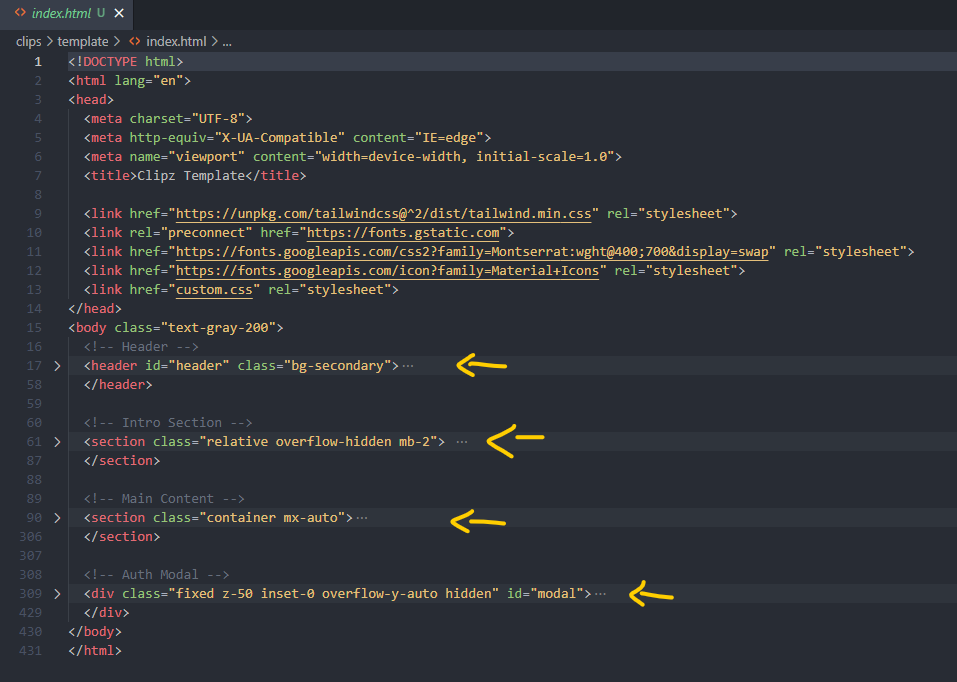
Open the **index.html** file, in the **template** directory. Inside this file, I've added dozens of comments.

These comments will help you navigate the sections of the project.

I will be referring to them whenever we need to work on a particular section.

I'm going to minimize some of the sections in the body tag.

**The home page has a total of four sections:**



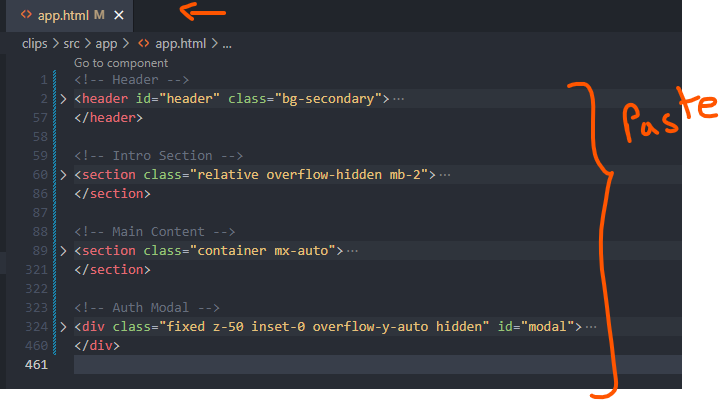
**We have the header, intro, main and authentication sections.**

Let's make a copy of them:



Next, open the **app.component.html** file:

We will replace the contents of this file with the code we've copied:



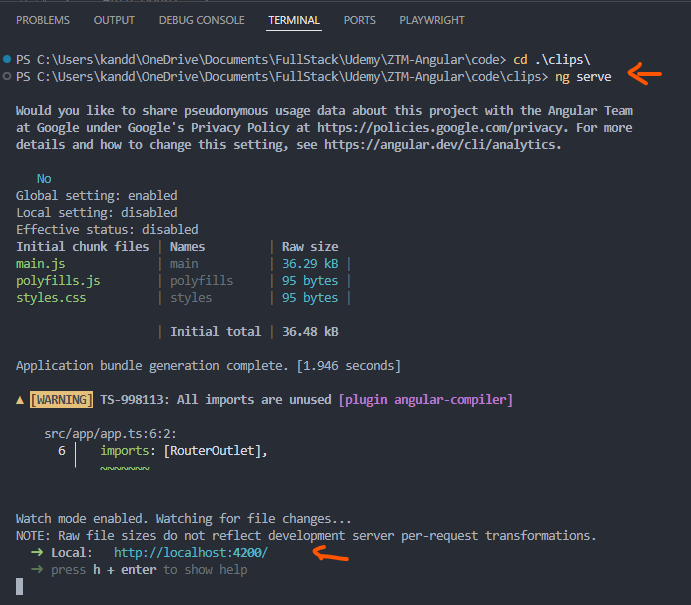
You may be thinking, why are we adding this code to the **index.html** file in the source directory?

We want to make this content dynamic. Therefore, we need to start a template in a component.

Otherwise, we won't be able to apply **pipes**, **directives** or **anything else** we need to make a template **dynamic**.

After pasting in the contents.

Let's turn on the server with the **ng serve** command.



Next, let's view the app in the browser:

The app is completely broken. It looks nothing like the static design:



That's to be expected since we didn't load the CSS for this template.

The template uses a framework called **Tailwind**, which requires a special installation process.

In the next lecture we will dive into **Tailwind**.