Next.js 15 app

1) Create the project

```
# Node 18.18+ recommended (Node 20 LTS is great)
npx create-next-app@latest my-next15-app \
  --ts --eslint --tailwind --app --src-dir \
  --import-alias "@/*"
```

- `--app` uses the App Router (default in v15).
- `--ts` gives you TypeScript out of the box (you can drop it if you want JS).
  Docs confirm `create-next-app@latest` and options.
  Next 15 is current, aligns with React 19 (App Router) and supports React 18 via Pages Router if needed.

2) Run it

```
cd my-next15-app
npm run dev
```

This starts the dev server (Turbopack-powered in dev on v15). Open the URL it prints (usually http://localhost:3000)

3) What you'll see / edit first

Your homepage lives at `app/page.tsx`. Edit it and your browser hot-refreshes.

Example minimal page:

```
// app/page.tsx
export default function Home() {
  return (
    <main style={{ padding: 24 }}>
      <h1>Hello Next.js 15</h1>
      <p>React + App Router is ready.</p>
    </main>
  );
}
```

4) Useful scripts

```
npm run dev      # develop (fast refresh)
npm run build    # production build
npm start        # run the built app
npm run lint     # lint (note: see tip below)
```

5) Tailwind is already wired up

You'll have `tailwind.config.ts` and `globals.css` set. Use classes right away:

```
<h1 className="text-3xl font-bold">Hello Next 15</h1>
```

(Enabled by the `--tailwind` flag in step 1.)

6) Node version tip

If you see a Node version error, switch to **≥ 18.17/18.18** (Node 20 LTS recommended). Tools like **nvm** make this easy:

```
nvm install 20
nvm use 20
```

(Node requirement and common fix referenced here.)

7) Quick notes about v15 / v15.5

- **React 19** features are used with the App Router; Pages Router can stay on React 18 if you need that path.
- **15.5** adds stronger **TypeScript** support and Turbopack improvements. If you're on 15.5+, you'll see those benefits out of the box.

Let's turn your fresh Next 15 app into a tiny multi-page site using the **App Router**. Copy these files into your project (create folders as needed).

1) Add a shared layout with nav

app/layout.tsx

```
import "./globals.css";
import Link from "next/link";

export const metadata = {
  title: "My Next 15 App",
  description: "Demo multi-page app",
};

export default function RootLayout({ children }:
{ children: React.ReactNode }) {
  return (
    <html lang="en">
      <body className="min-h-screen">
        <header className="border-b">
          <nav className="mx-auto max-w-5xl px-4 py-3 flex
gap-4">
            <Link href="/" className="font-medium">Home</
Link>
            <Link href="/about">About</Link>
            <Link href="/contact">Contact</Link>
            <Link href="/shows">Shows</Link>
            <Link href="/faq">FAQ</Link>
          </nav>
        </header>
        <main className="mx-auto max-w-5xl px-4
py-8">{children}</main>
        <footer className="mt-12 border-t py-6 text-sm
text-gray-500 text-center">
          © {new Date().getFullYear()} My Next 15 App
        </footer>
      </body>
    </html>
  );
}
```

2) Home page (replace your existing `app/page.tsx`)

app/page.tsx

```
export default function Home() {
  return (
    <>
      <h1 className="text-3xl font-bold mb-2">Hello Next.js
15</h1>
      <p className="text-gray-600">Now with multiple
pages.</p>
```

```
    </>
  );
}
```

3) Static pages

app/about/page.tsx

```
export const metadata = { title: "About" };

export default function AboutPage() {
  return (
    <>
      <h1 className="text-2xl font-semibold mb-3">About</
h1>
      <p>We’re building a delightful React + Next 15 app.</
p>
    </>
  );
}
```

app/contact/page.tsx

```
export const metadata = { title: "Contact" };

export default function ContactPage() {
  return (
    <>
      <h1 className="text-2xl font-semibold mb-3">Contact</
h1>
      <form className="grid gap-3 max-w-md">
        <label className="grid gap-1">
          <span>Name</span>
          <input className="border px-3 py-2 rounded"
placeholder="Jane Doe" />
        </label>
        <label className="grid gap-1">
          <span>Email</span>
          <input type="email" className="border px-3 py-2
rounded" placeholder="jane@example.com" />
        </label>
        <label className="grid gap-1">
          <span>Message</span>
          <textarea className="border px-3 py-2 rounded"
rows={4} />
        </label>
        <button className="border px-4 py-2 rounded
hover:bg-gray-50 w-fit">Send</button>
      </form>
    </>
  );
}
```

4) A list page + dynamic routes

app/shows/page.tsx

```
import Link from "next/link";

const shows = [
  { slug: "hms-pinafore", title: "H.M.S. Pinafore", date:
"2025-10-03" },
  { slug: "mikado", title: "The Mikado", date: "2025-12-12"
},
  { slug: "gondoliers", title: "The Gondoliers", date:
"2026-02-20" },
];

export const metadata = { title: "Shows" };

export default function ShowsPage() {
  return (
    <>
      <h1 className="text-2xl font-semibold mb-4">Upcoming
Shows</h1>
      <ul className="grid gap-3">
        {shows.map((s) => (
          <li key={s.slug} className="border rounded p-4
flex items-center justify-between">
            <div>
              <div className="font-medium">{s.title}</div>
              <div className="text-sm text-gray-600">{new
Date(s.date).toLocaleDateString()}</div>
            </div>
            <Link href={`/shows/${s.slug}`}
className="underline">Details</Link>
          </li>
        ))}
      </ul>
    </>
  );
}
```
app/shows/[slug]/page.tsx

```
type Params = { slug: string };

const data = {
  "hms-pinafore": { title: "H.M.S. Pinafore", synopsis: "A
nautical comic opera." },
  "mikado": { title: "The Mikado", synopsis: "A satire set
in Titipu." },
  "gondoliers": { title: "The Gondoliers", synopsis:
"Venetian romance and mistaken identity." },
};

export async function generateStaticParams() {
  return Object.keys(data).map((slug) => ({ slug }));
}

export function generateMetadata({ params }: { params:
Params }) {
  const show = data[params.slug];
  return { title: show ? show.title : "Show" };
}

export default function ShowDetail({ params }: { params:
Params }) {
  const show = data[params.slug];
  if (!show) return <div>Show not found.</div>;
  return (
    <>
      <h1 className="text-2xl font-semibold
mb-3">{show.title}</h1>
      <p>{show.synopsis}</p>
    </>
  );
}
```

5) A simple FAQ using a route group (clean URL, organized files)

Create a route group so the folder name doesn’t appear in the URL.

app/(marketing)/faq/page.tsx

```
export const metadata = { title: "FAQ" };

export default function FAQPage() {
  return (
    <>
      <h1 className="text-2xl font-semibold mb-3">FAQ</h1>
      <details className="mb-3">
        <summary className="cursor-pointer font-
medium">What is this?</summary>
        <p className="mt-2">A minimal multi-page Next.js 15
demo.</p>
      </details>
      <details>
        <summary className="cursor-pointer font-medium">Is
React included?</summary>
        <p className="mt-2">Yep — App Router runs on
React.</p>
      </details>
    </>
  );
}
```

6) Loading and 404 states (nice UX)

app/loading.tsx

```
export default function Loading() {
  return <div className="animate-pulse">Loading…</div>;
}
```

app/not-found.tsx

```
import Link from "next/link";

export default function NotFound() {
  return (
    <div className="grid gap-2">
      <h1 className="text-2xl font-semibold">404 — Not
found</h1>
      <p>We couldn’t find that page.</p>
      <Link href="/" className="underline">Go home</Link>
    </div>
  );
}
```

```
npm run dev
```

Then visit:

- `/` Home
- `/about`
- `/contact`
- `/shows` and `/shows/hms-pinafore` (or another slug)
- `/faq`
- `/api/hello` (returns JSON)

In **Next.js App Router**:

- **`[slug]`** → you literally name the folder with square brackets.
  Example:

```
app/
```

```
  shows/
    [slug]/
      page.tsx
```

That makes `/shows/hms-pinafore` or `/shows/mikado` work dynamically.
`(marketing)` → you literally name the folder with parentheses.

```
app/

  (marketing)/
    faq/
      page.tsx
```

Route groups like `(marketing)` **don't appear in the URL**.
So the URL is just `/faq`, not `/(marketing)/faq`.
Actually type `[slug]` and `(marketing)` in your folder names — those characters have meaning in Next.js.

`(marketing)` isn't a keyword in Next.js, it's just an **example name** for a *route group*.

What a Route Group is

- A **route group** is a folder wrapped in parentheses (e.g. `(marketing)` or `(app)`).
- Next.js **ignores the group name in the URL**, but still uses the folder to organize your code.

Why use it?

Imagine you have two sets of pages:

- **Marketing site** → public-facing pages like `/about`, `/faq`, `/contact`.
- **App/dashboard** → private pages like `/dashboard`, `/settings`.

You don't want `/marketing/about` in the URL — you just want `/about`.
So you put them in a route group:

```
app/
  (marketing)/
```

```
    about/page.tsx     → /about
    faq/page.tsx       → /faq
    contact/page.tsx   → /contact
  (dashboard)/
    dashboard/page.tsx → /dashboard
    settings/page.tsx  → /settings
```
Here:

- `(marketing)` and `(dashboard)` are **purely organizational**.
- They keep files grouped, but don't affect the routes.

Next.js **App Router** (which you're using in Next 15) mixes **SSR (Server-Side Rendering)** and **hydration** by default, but the details depend on the component type.