



JUNE 13 & 17, 2025 IN AMSTERDAM



NOVEMBER 18 & 21, 2025 IN NEW YORK

# THE BIGGEST REACT CONFERENCE WORLDWIDE

10TH ANNIVERSARY

2

Tracks: Base Camp &  
Summit

60+

Speakers sharing  
latest insights

10K+

Devs from all over  
the globe

1500

Luckies meet in  
Amsterdam

[STREAM PAGE](#)

[RECORDINGS AT GITNATION.COM](#)

# Next.js 15 Development Bootcamp

Maurice de Beijer  
@mauricedb



- Maurice de Beijer
- The Problem Solver
- Freelance developer/instructor
- Twitter: [@mauricedb](https://twitter.com/mauricedb)
- Web: <https://www.theproblemsolver.dev/>
- E-mail: [maurice.de.beijer@gmail.com](mailto:maurice.de.beijer@gmail.com)



# Topics

- **Why** use Next.js
- Creating a **new Next.js** application
- Generating a **landing page** using Vercel V0
- Adding a **Postgres SQL database** using Docker
- Adding the **Prisma ORM** with the database schema
- **Seeding** the database with data
- Adding **client-side interactions and state**
- **Deploying** the application to Vercel

Type it out  
by hand?

*"Typing it drills it into your brain much better than simply copying and pasting it. You're forming new neuron pathways. Those pathways are going to help you in the future. Help them out now!"*

# Prerequisites

Install Node, NPM & Docker

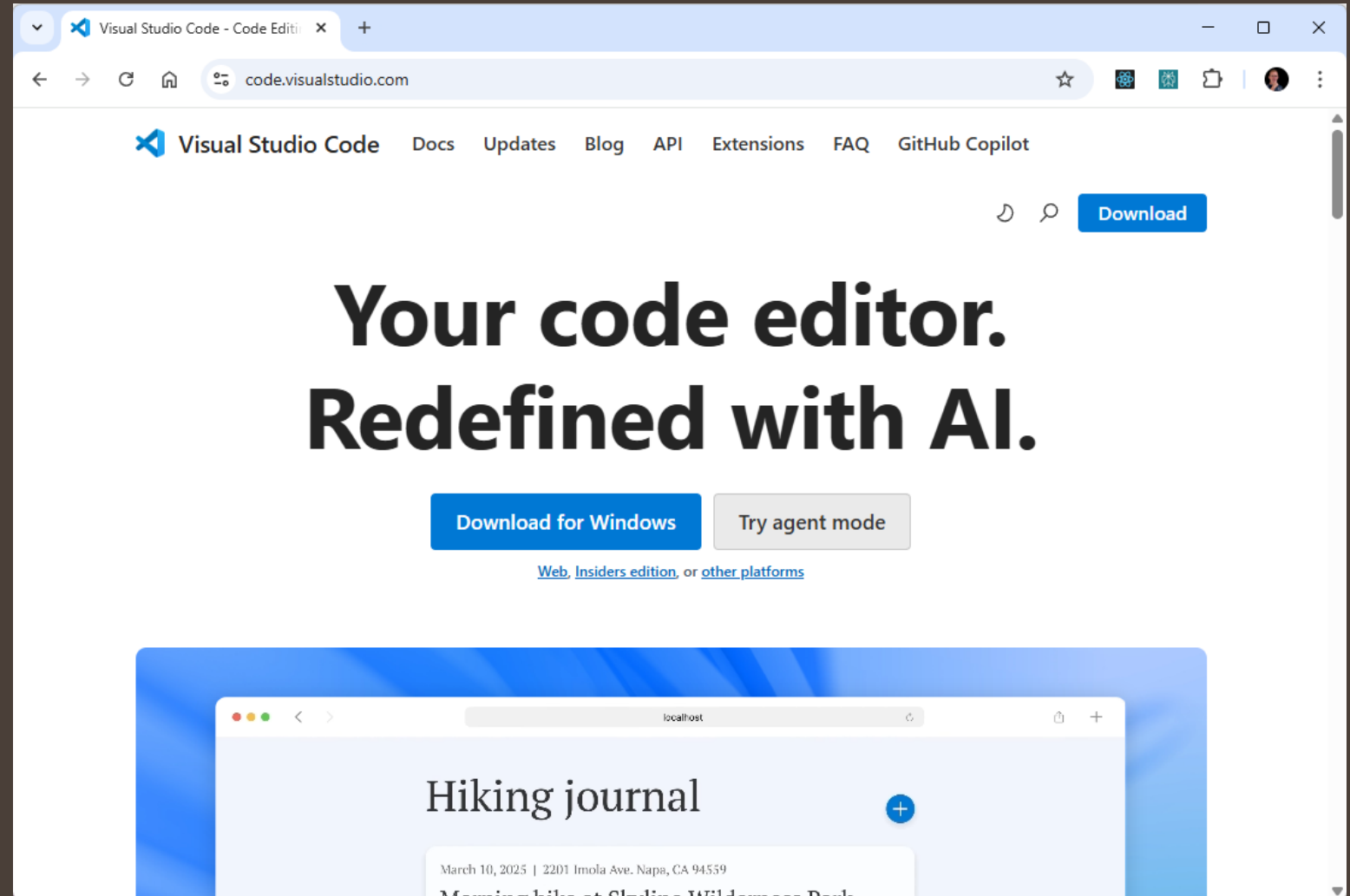
# Install Node.js & NPM

The image displays four screenshots related to the installation of development tools:

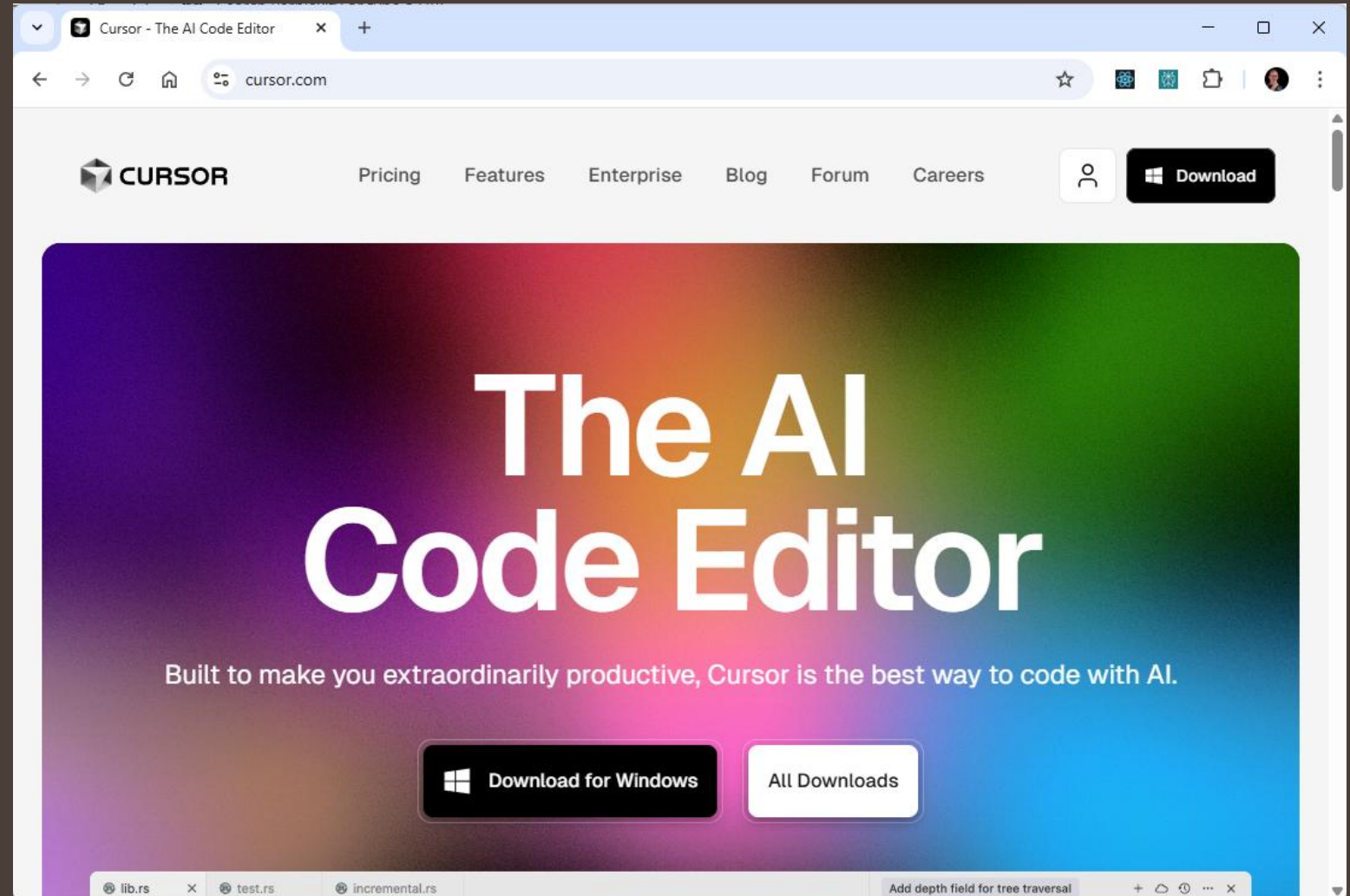
- Node.js Website:** The Node.js homepage (nodejs.org/en/) featuring the headline "Run JavaScript Everywhere". It describes Node.js as a free, open-source, cross-platform JavaScript runtime environment. A prominent green button labeled "Download Node.js (LTS)" is visible. A code snippet for a simple HTTP server is shown on the right.
- Git Downloads Page:** The Git website (git-scm.com/downloads) with the headline "git --distributed-even-if-your-workflow-isnt". It offers download links for macOS, Windows, and Linux/Unix. A monitor graphic displays the "Latest source Release 2.38.0" with a "Download for Windows" button.
- Docker Desktop Docs:** The Docker Desktop documentation page (docs.docker.com/get-started/get-docker/). It provides instructions for installing Docker Desktop on Mac, Windows, and Linux, describing it as a native application that delivers all Docker tools.
- Windows PowerShell Terminal:** A screenshot of a Windows PowerShell terminal window showing the output of version commands:

```
PS C:\demos> node --version
v22.16.0
PS C:\demos> git --version
git version 2.45.2.windows.1
PS C:\demos> npm --version
10.2.4
PS C:\demos> docker --version
Docker version 27.3.1, build ce12230
PS C:\demos> |
```

VS Code



Or Cursor



# Following Along



```
File Edit Selection View Go Run Terminal Help
package.json docker-compose.yml X
docker-compose.yml
1 services:
2   postgres:
3     image: postgres:17-alpine
4     container_name: Movie-Comparer-Postgres
5     environment:
6       POSTGRES_DB: postgres
7       POSTGRES_USER: postgres
8       POSTGRES_PASSWORD: postgres
9     ports:
10      - '54323:5432'
11     volumes:
12      - postgres_data:/var/lib/postgresql/data
13
14 volumes:
15   postgres_data:
```

- Repo: <https://github.com/mauricedb/next-15-bootcamp>
- Slides: <https://www.theproblemsolver.dev/docs/next-15-bootcamp.pdf>

# The changes



Commits · mauricedb/next-15-  
github.com/mauricedb/next-15-bootcamp/com...

mauricedb / next-15-bootcamp

<> Code Issues Pull requests Actions Projects Wiki Settings

## Commits

main All users All time

Commits on Jun 12, 2025

- Fix HTML entity in movie night message for proper rendering**  
mauricedb committed 2 minutes ago · 1 / 1 4d47072
- Disabling /compare when no movies are selected**  
mauricedb committed 2 minutes ago ab8576f
- Comparing movies**  
mauricedb committed 2 minutes ago 1a411a6
- Adding client-side interactions and state**  
mauricedb committed 2 minutes ago 2d87c57
- Comparing movies from V0.dev**  
mauricedb committed 2 minutes ago 8dd5aea
- Rendering movie cards**  
mauricedb committed 2 minutes ago 3030d38
- Seeding the database**  
mauricedb committed 11 minutes ago bd178f8
- Adding the database schema**  
mauricedb committed 11 minutes ago 7acef17
- Adding the Prisma ORM**  
mauricedb committed 11 minutes ago f89d9a6
- Adding a Postgres SQL database**  
mauricedb committed 11 minutes ago 7e11886
- Using shared layouts

# The application




[Home](#) [Movies](#) [Compare](#) [Login](#)

## Movie Comparison


Compare movies side by side to help you decide what to watch next. Toggle off movies you're not interested in.

☐ Consider



**Dune: Part Two**  
★ 8.146 6,702 25.8  
February 27, 2024  
**Genres**  
Adventure Science Fiction  
**Director**  
Denis Villeneuve  
**Main Cast**  
Timothée Chalamet, Florence Pugh, Rebecca Ferguson, Austin Butler +1 more  
**Overview**  
Follow the mythic journey of Paul Atreides as he unites with Chani and the Fremen while on a path of revenge against the conspirators who destroyed his family. Facing a choice between the love of his life and the fate of the known universe, Paul endeavors to prevent a terrible future only he can foresee.

☐ Consider



**Interstellar**  
★ 8.5 37,212 36.7  
November 5, 2014  
**Genres**  
Adventure Drama Science Fiction  
**Director**  
Christopher Nolan  
**Main Cast**  
Michael Caine, Timothée Chalamet, Anne Hathaway, Matt Damon +1 more  
**Overview**  
The adventures of a group of explorers who make use of a newly discovered wormhole to surpass the limitations on human space travel and conquer the vast distances involved in an interstellar voyage.

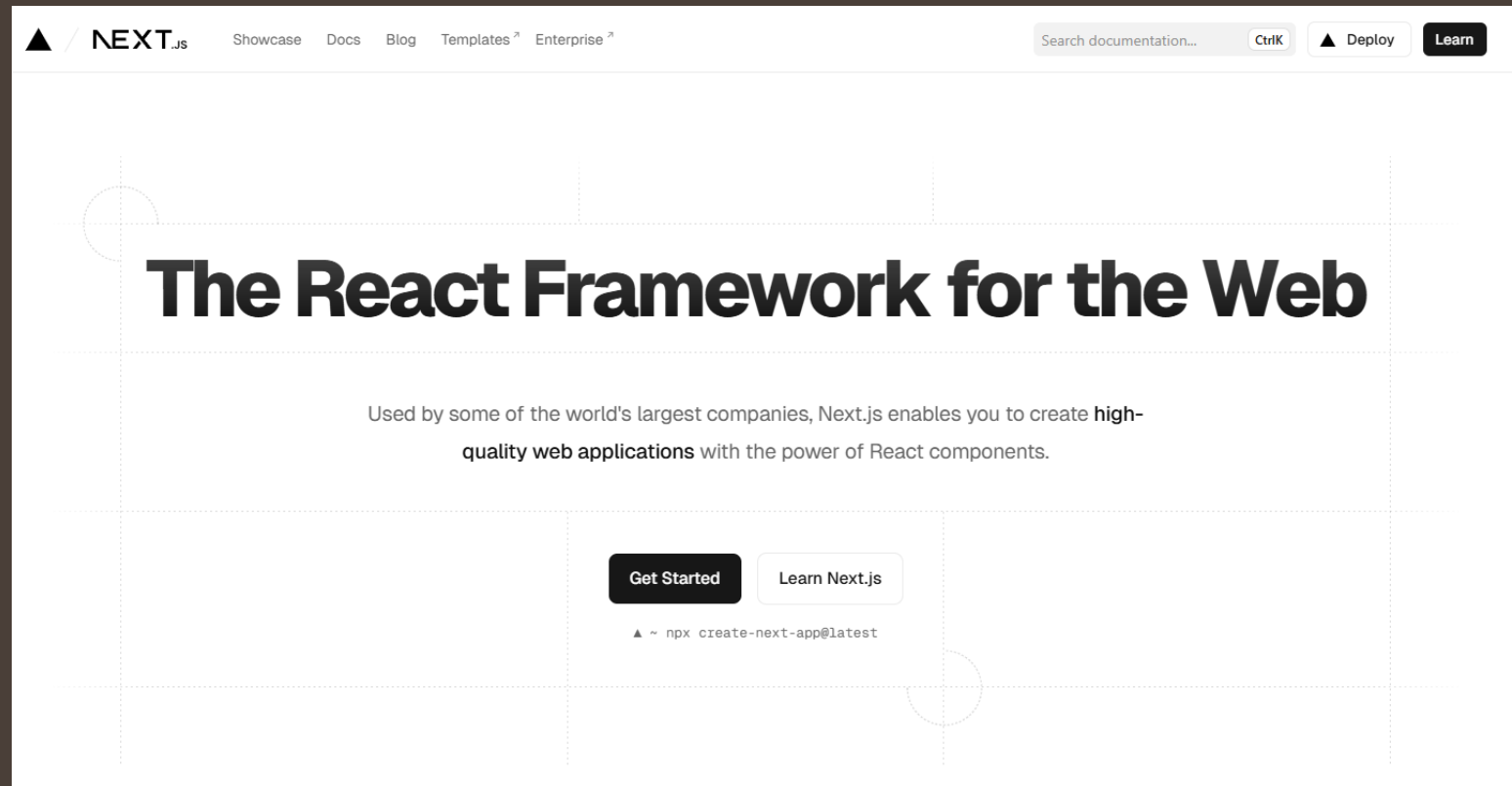
☐ Consider




**Inception**  
★ 8.368 37,515 22.4  
July 15, 2010  
**Genres**  
Action Adventure Science Fiction  
**Director**  
Christopher Nolan  
**Main Cast**  
Michael Caine, Tom Hardy, Leonardo DiCaprio, Elliot Page +1 more  
**Overview**  
Cobb, a skilled thief who commits corporate espionage by infiltrating the subconscious of his targets is offered a chance to regain his old life as payment for a task considered to be impossible: "inception", the implantation of another person's idea into a target's subconscious.

# Why use Next.js 15

# Why use Next.js



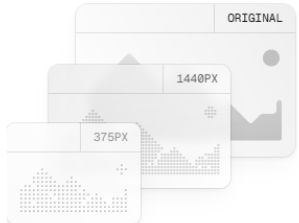
# Why use Next.js

 / **NEXT.js** [Showcase](#) [Docs](#) [Blog](#) [Templates](#) [Enterprise](#)

Search documentation... [CtrlK](#) [Deploy](#) [Learn](#)


## What's in Next.js?

Everything you need to build great products on the web.



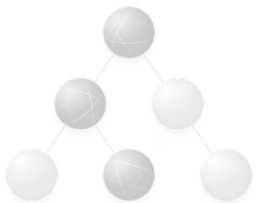
### Built-in Optimizations

Automatic Image, Font, and Script Optimizations for improved UX and Core Web Vitals.



### Dynamic HTML Streaming

Instantly stream UI from the server, integrated with the App Router and React Suspense.



### React Server Components

Add components without sending additional client-side JavaScript. Built on the latest React features.

### Data Fetching

Make your React component async and await your data. Next.js supports both server and client data fetching.

### CSS Support

Style your application with your favorite tools, including support for CSS Modules, Tailwind CSS, and popular community libraries.

### Client and Server Rendering

Flexible rendering and caching options, including Incremental Static Regeneration (ISR), on a per-page level.

### Server Actions

Run server code by calling a function. Skip the API. Then, easily revalidate cached data and update your UI in one network roundtrip.

### Route Handlers

Build API endpoints to securely connect with third-party services for handling auth or listening for webhooks.

### Advanced Routing & Nested Layouts

Create routes using the file system, including support for more advanced routing patterns and UI layouts.

### Middleware

Take control of the incoming request. Use code to define routing and access rules for authentication, experimentation, and internationalization.

### Next.js 15


The power of full-stack to the frontend. [Read the release notes.](#)

# Why use Next.js

- **React 19 Support**
  - It provides full compatibility with React 19, giving you access to the latest React features like Server Components improvements, enhanced Suspense behavior, and better concurrent rendering capabilities.
- **React Server Components**
  - Server Components have been refined with better streaming support, improved error boundaries, and more predictable behavior, making it easier to build applications that render efficiently on the server.
- **Great Developer Experience**
  - The development server is faster with better hot module replacement. The framework includes enhanced error messages, better debugging tools, and improved TypeScript integration that makes development more productive.

# Why use Next.js

- **Enhanced App Router**
  - The App Router continues to mature with better performance, more stable APIs, and improved data fetching patterns. This gives you more flexibility in how you structure and render your applications.
- **Performance Improvements**
  - Next.js 15 introduces significant performance optimizations, including better bundle splitting, improved tree shaking, and enhanced caching mechanisms.
  - It supports partial prerendering (PPR) as a stable feature, allowing you to combine static and dynamic content more efficiently.
- **Turbopack Integration**
  - Next.js 15 includes better integration with Turbopack (Webpack's successor), providing faster build times and more efficient bundling, especially for larger applications.



# Creating a new Next.js application

## Creating a new Next.js application

- **Creating a new project** using a wizard with a questionnaire
  - `npx create-next-app@latest`

## Creating a new Next.js application

- **Creating a project based on a Vercel Next.js example**
  - `npx create-next-app@latest my-app1 --example with-styled-components`
  - See <https://github.com/vercel/next.js/tree/main/examples>
- **Create a new project based on another example**
  - `npx create-next-app@latest my-app1 --example https://github.com/mui/material-ui/tree/master/examples/material-ui-nextjs-ts`

# Creating a new Shadcn/Next.js application

- When **using Shadcn/ui** there is a shortcut
  - `npx shadcn@latest init`

# Creating a new Next.js application

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

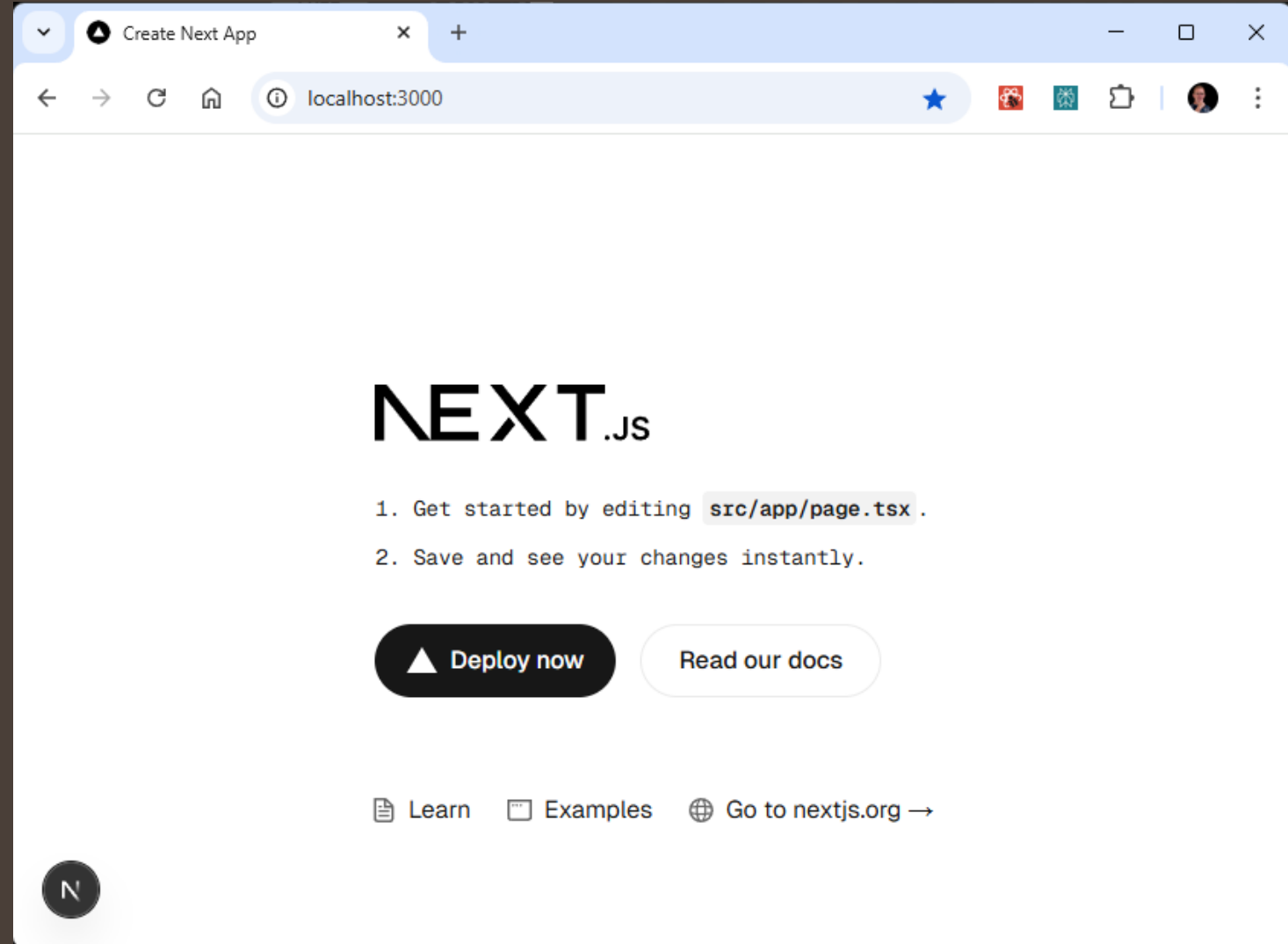
PS C:\Repos> cd C:\demos\
PS C:\demos>
PS C:\demos> npx create-next-app@latest
Need to install the following packages:
create-next-app@15.3.2
Ok to proceed? (y)
✓ What is your project named? ... next-15-bootcamp
✓ Would you like to use TypeScript? ... No / Yes
✓ Would you like to use ESLint? ... No / Yes
✓ Would you like to use Tailwind CSS? ... No / Yes
✓ Would you like your code inside a 'src/' directory? ... No / Yes
✓ Would you like to use App Router? (recommended) ... No / Yes
✓ Would you like to use Turbopack for 'next dev'? ... No / Yes
✓ Would you like to customize the import alias ('@/*' by default)? ... No / Yes
Creating a new Next.js app in C:\demos\next-15-bootcamp.

Using npm.

Initializing project with template: app-tw

Installing dependencies:
- react
- react-dom
- next
```

# Creating a new Next.js application

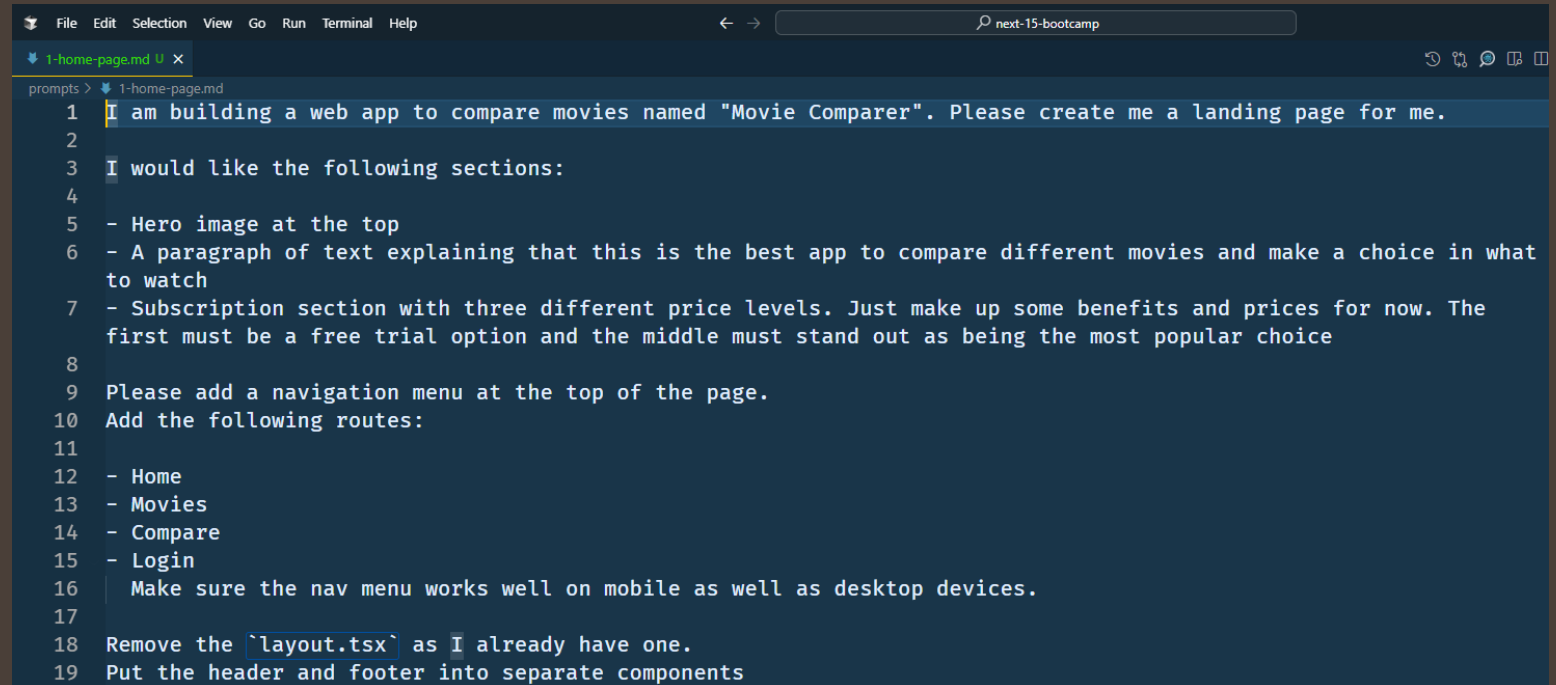


# Generating a landing page using Vercel V0

# Generating a landing page using Vercel V0

- **Rapid prototyping with complete projects**
  - Generates full Next.js applications with proper folder structure, configuration, and git setup that you can immediately download and run locally
- **Modern Next.js best practices built-in**
  - Uses the latest App Router, Server Actions, TypeScript, and Tailwind CSS automatically, keeping your code current with framework standards
- **Real-time interactive development**
  - Preview and test your application instantly in the browser while making changes through natural language prompts, no local setup required
- **Production-ready code quality**
  - Generates clean, well-organized code following industry best practices rather than throwaway prototypes, with integrated shadcn/ui components
- **Accessible to non-developers**
  - Enables designers, product managers, and entrepreneurs to build functional web applications using plain English descriptions rather than requiring deep coding knowledge

# The Prompt



The screenshot shows a code editor with a dark theme. The menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The search bar contains 'next-15-bootcamp'. The file explorer on the left shows '1-home-page.md'. The main editor area contains a prompt for a web app, with line numbers 1 through 19 on the left. The prompt text is as follows:

```
1 I am building a web app to compare movies named "Movie Comparer". Please create me a landing page for me.
2
3 I would like the following sections:
4
5 - Hero image at the top
6 - A paragraph of text explaining that this is the best app to compare different movies and make a choice in what
  to watch
7 - Subscription section with three different price levels. Just make up some benefits and prices for now. The
  first must be a free trial option and the middle must stand out as being the most popular choice
8
9 Please add a navigation menu at the top of the page.
10 Add the following routes:
11
12 - Home
13 - Movies
14 - Compare
15 - Login
16 Make sure the nav menu works well on mobile as well as desktop devices.
17
18 Remove the `layout.tsx` as I already have one.
19 Put the header and footer into separate components
```

# Generating a landing page using Vercel V0

The screenshot displays the Vercel V0 interface, which is used for generating web content. On the left, a text input area contains the prompt: "I am building a web app to compare movies named 'Movie Comparer'. Please create me a landing page for me. I would like the following sections: Hero image at the top, A paragraph of text explaining that this is the best app to compare different movies and make a choice in what to watch, Subscription section with three different price levels. Just make up some benefits and prices for now. The first must be a free trial option and the middle must stand out as being the most popular choice. Please add a navigation menu at the top of the page. Add the following routes: Home, Movies, Compare, Login. Make sure the nav menu works well on mobile as well as desktop devices." Below the input, it shows "Thought for 6 seconds" and the generated output titled "Movie Comparer Landing Page". The output includes a preview of the landing page and a list of suggestions for further customization, such as "Add Integration" and "Add movie comparison feature". A warning message states "You are running low on credits. Upgrade Plan". At the bottom, there is a console area and a disclaimer: "V0 may make mistakes. Please use with discretion."

**Movie Comparer**

Home Movies Compare Login

## Compare. Decide. Watch.

Find your next favorite movie with the most powerful comparison tool

[Start Comparing Now](#)

### The Ultimate Movie Comparison Experience

Movie Comparer is the best app to compare different movies and make an informed choice on what to watch next. Our platform allows you to compare movies side by side based on ratings, reviews, cast, runtime, and dozens of other factors. Never waste time on a disappointing movie again! With our advanced comparison tools, you'll always make the perfect choice for movie night.

# Generating a landing page using Vercel V0

PersonalFree

Movie Comparer landing page

Private

I am building a web app to compare movies named "Movie Comparer". Please create me a landing page for me.

I would like the following sections:

Hero image at the top

A paragraph of text explaining that this is the best app to compare different movies and make a choice in what to watch

Subscription section with three different price levels. Just make up some benefits and prices for now. The first must be a free trial option and the middle must stand out as being the most popular choice

Please add a navigation menu at the top of the page. Add the following routes:

Home

Movies

Compare

Login Make sure the nav menu works well on mobile as well as desktop devices.

Thought for 6 seconds

### Movie Comparer Landing Page

I'll create a responsive landing page for your "Movie Comparer" web app with all the sections you requested.

Version 1

Latest

Viewing

app/page.tsx

Generated

Suggestions

Add Integration

Add movie comparison feature

You are running low on credits.

Upgrade Plan

Ask a follow up...

v0-1.5-md

v0 may make mistakes. Please use with discretion.

>>

Preview

Code

app

compare

login

movies

layout.tsx

page.tsx

page.tsx

+9

page.tsx

+45

page.tsx

+9

layout.tsx

+25

page.tsx

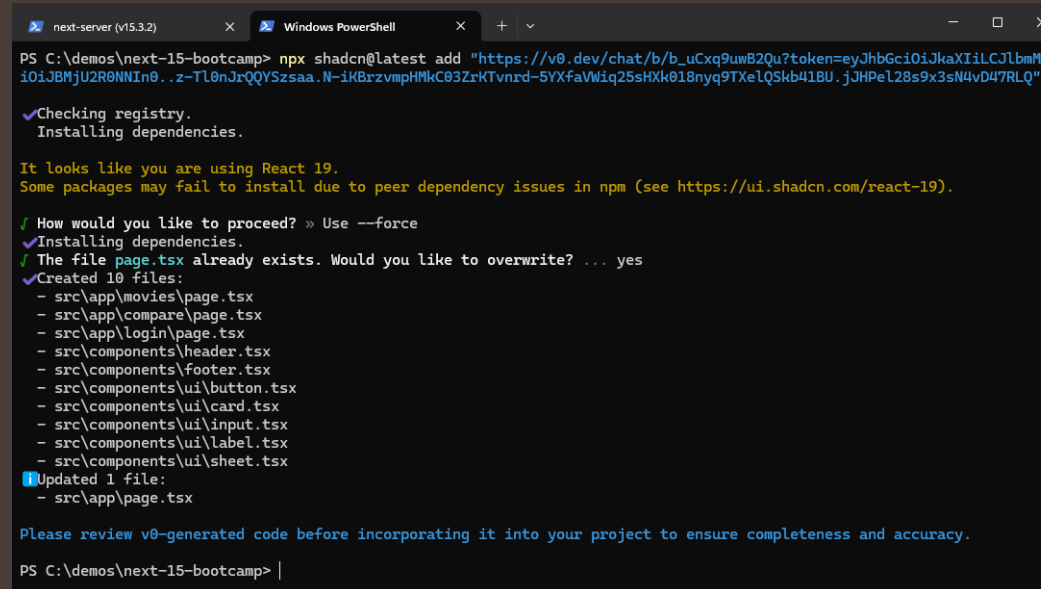
+235

app > layout.tsx

```
1 import type React from "react"
2 import "@app/globals.css"
3 import { ThemeProvider } from "@components/theme-provider"
4
5 export const metadata = {
6   title: "Movie Comparer - Compare Movies and Make the Right Choice",
7   description: "Compare movies side by side and find your next favorite film with Movie Comparer",
8 }
9
10 export default function RootLayout({
11   children,
12 }: {
13   children: React.ReactNode
14 }) {
15   return (
16     <html lang="en" suppressHydrationWarning>
17       <body>
18         <ThemeProvider attribute="class" defaultTheme="system" enableSystem disableTransitionOnChange>
19           {children}
20         </ThemeProvider>
21       </body>
22     </html>
23   )
24 }
25
```

# Add to codebase

- `npx shadcn@latest add "https://vo.dev/chat/b/b_uCxq9uwB2Qu"`



```
next-server (v15.3.2) | Windows PowerShell
PS C:\demos\next-15-bootcamp> npx shadcn@latest add "https://vo.dev/chat/b/b_uCxq9uwB2Qu?token=eyJhbGciOiJIcXkiLCJlbnMiOiJBMjU2R0NNIn0..z-TL0nJrQQYSzsaa.N-ikBrzvmpHMKC03ZrKTvnrd-SYXfaVWiq25sHXk018nyq9TXelQSkb41BU.jJHPeL28s9x3sN4vD47RLQ"

✓Checking registry.
  Installing dependencies.

It looks like you are using React 19.
Some packages may fail to install due to peer dependency issues in npm (see https://ui.shadcn.com/react-19).

✓ How would you like to proceed? » Use --force
✓Installing dependencies.
✓The file page.tsx already exists. Would you like to overwrite? ... yes
✓Created 10 files:
- src\app\movies\page.tsx
- src\app\compare\page.tsx
- src\app\login\page.tsx
- src\components\header.tsx
- src\components\footer.tsx
- src\components\ui\button.tsx
- src\components\ui\card.tsx
- src\components\ui\input.tsx
- src\components\ui\label.tsx
- src\components\ui\sheet.tsx
■Updated 1 file:
- src\app\page.tsx

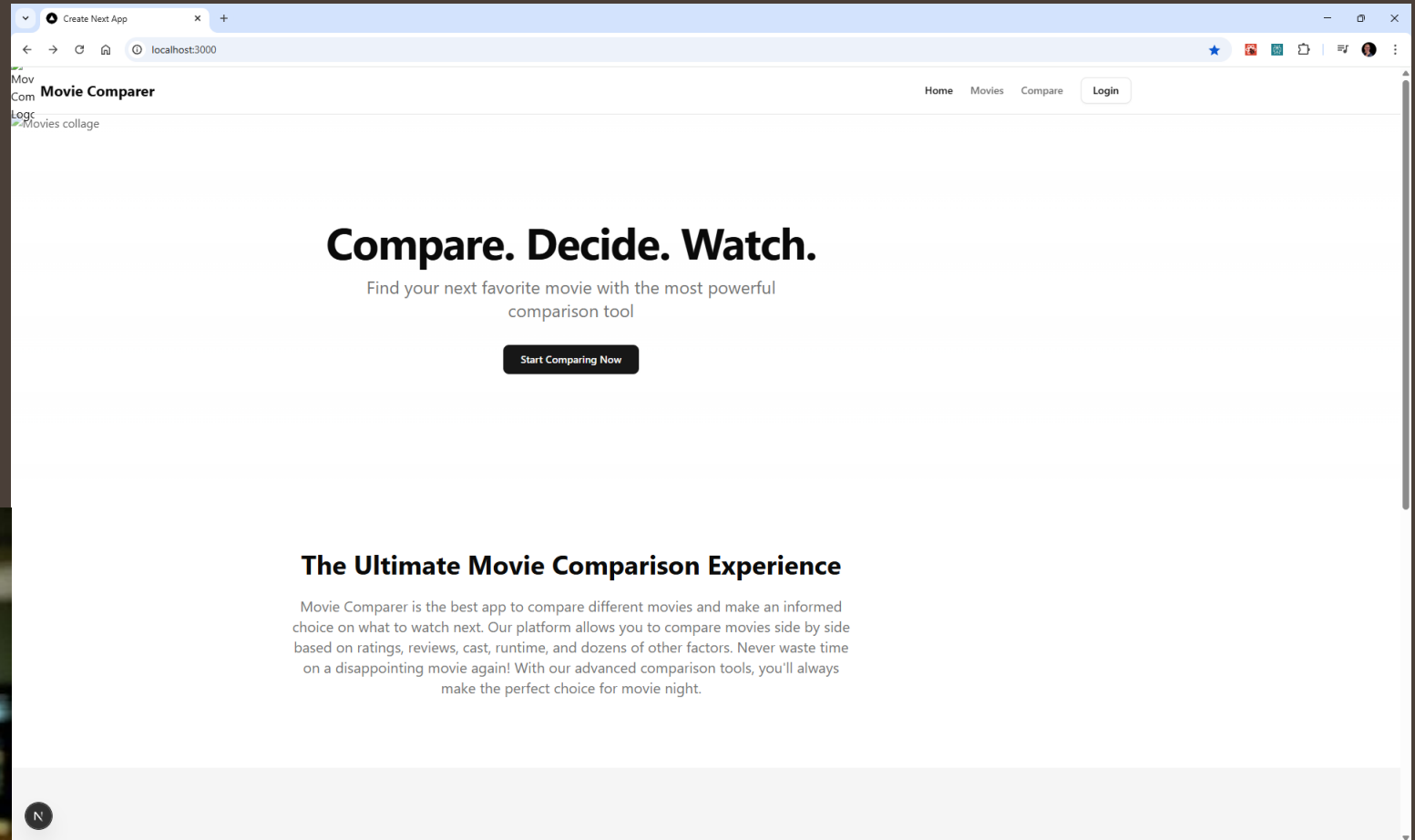
Please review v0-generated code before incorporating it into your project to ensure completeness and accuracy.

PS C:\demos\next-15-bootcamp> |
```

# Placeholder image

- Not: the placeholder image is not added to the project
  - <https://github.com/mauricedb/next-15-bootcamp/blob/main/public/placeholder.svg>

Generating a  
landing page  
using Vercel V0





Using shared layouts

# Using shared layouts

- **Consistent UI structure across pages**
  - Layout components allow you to define shared elements like headers, navigation, and footers once and automatically apply them to multiple pages, ensuring a cohesive user experience throughout your application.
- **Reduced code duplication**
  - Instead of importing and wrapping the same layout components in every page file, you can define the layout once and have it automatically wrap all pages in that route segment, leading to cleaner and more maintainable code.
- **Improved performance through persistent state**
  - Layout components remain mounted when navigating between pages that share the same layout, preserving component state and avoiding unnecessary re-renders of common UI elements like navigation menus or sidebars.
- **Hierarchical layout composition**
  - The App Router's nested layout system allows you to create sophisticated layout hierarchies where different route segments can have their own layouts that compose together, enabling complex page structures with minimal code.
- **Better SEO and metadata management**
  - Layouts provide a centralized place to define page metadata, title templates, and other SEO-related elements that should be consistent across sections of your site, making it easier to maintain proper search engine optimization.

# Using shared layouts

```
File Edit Selection View Go Run Terminal Help
next-15-bootcamp
layout.tsx M x page.tsx 1, M
src > app > layout.tsx > ...
16
17 export const metadata: Metadata = {
18   title: 'Movie Comparer - Compare Movies and Make the Right Choice',
19   description:
20     'Compare movies side by side and find your next favorite film with Movie Comparer',
21 };
22
23 export default function RootLayout({
24   children,
25 }: Readonly<{
26   children: React.ReactNode;
27 }>) {
28   return (
29     <html lang="en">
30       <body
31         className={` ${geistSans.variable} ${geistMono.variable} antialiased`>
32         >
33         <div className="flex flex-col min-h-screen container mx-auto">
34           <Header />
35
36           <main className="flex-1">{children}</main>
37
38           <Footer />
39         </div>
40       </body>
41     </html>
42   );
43 }
```

# Using shared layouts



## Movie Comparer

[Home](#) [Movies](#) [Compare](#) [Login](#)

### Login


Enter your email and password to access your account

Email

Password [Forgot password?](#)

[Login](#)

Don't have an account? [Sign up](#)



Movie Comparer

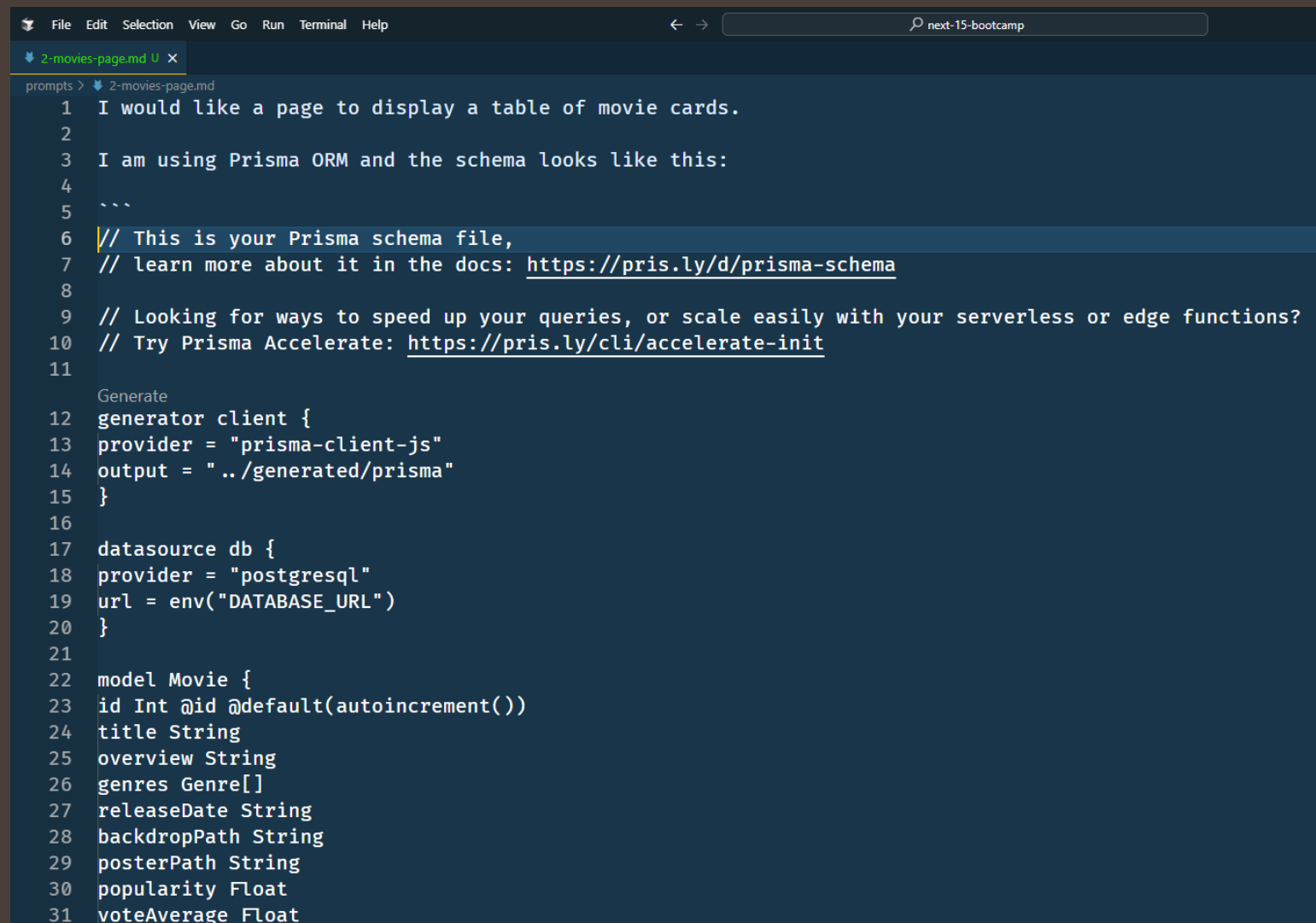
© 2025 Movie Comparer. All rights reserved.

# Rendering movie cards

# Rendering movie cards

- We want the user of the app to be able to pick movies to compare
  - We will use Vercel V0 again to generate the components
  - For now, we will use some hard coded data

# Rendering movie cards



The screenshot shows a code editor with a dark theme. At the top, there's a menu bar with 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help'. Below the menu bar, there's a search bar with the text 'next-15-bootcamp'. The main area of the editor displays a file named '2-movies-page.md'. The content of the file is a list of prompts for an AI model, numbered 1 through 31. The prompts describe a task to render movie cards using Prisma ORM and provide a Prisma schema file. The schema file is a Prisma schema for a database named 'db' with a 'Movie' model. The model has fields: 'id' (Int, @id, @default(autoincrement())), 'title' (String), 'overview' (String), 'genres' (Genre[]), 'releaseDate' (String), 'backdropPath' (String), 'posterPath' (String), 'popularity' (Float), and 'voteAverage' (Float).

```
File Edit Selection View Go Run Terminal Help
next-15-bootcamp
2-movies-page.md U X
prompts > 2-movies-page.md
1 I would like a page to display a table of movie cards.
2
3 I am using Prisma ORM and the schema looks like this:
4
5 ```
6 // This is your Prisma schema file,
7 // learn more about it in the docs: https://pris.ly/d/prisma-schema
8
9 // Looking for ways to speed up your queries, or scale easily with your serverless or edge functions?
10 // Try Prisma Accelerate: https://pris.ly/cli/accelerate-init
11
12 Generate
13 generator client {
14   provider = "prisma-client-js"
15   output = "../generated/prisma"
16 }
17
18 datasource db {
19   provider = "postgresql"
20   url = env("DATABASE_URL")
21 }
22
23 model Movie {
24   id Int @id @default(autoincrement())
25   title String
26   overview String
27   genres Genre[]
28   releaseDate String
29   backdropPath String
30   posterPath String
31   popularity Float
32   voteAverage Float
```

# Rendering movie cards

Personal Free / Movie card component Unlisted

Move the movie data into the page itself

Thought for 2 seconds

I'll move the movie data directly into the movies page and remove the separate data file.

Version 4 Latest Viewing

app/movies/page.tsx Edited

app/movies/data.ts Deleted

I've moved the movie data directly into the movies page file and removed the separate data.ts file. The movies array is now defined at the top of the page file, making the code more self-contained and easier to manage for this specific use case.

Suggestions

You are running low on credits. Upgrade Plan

Ask a follow-up...

v0-1.5-md


v0 may make mistakes. Please use with discretion.

Preview </> Code

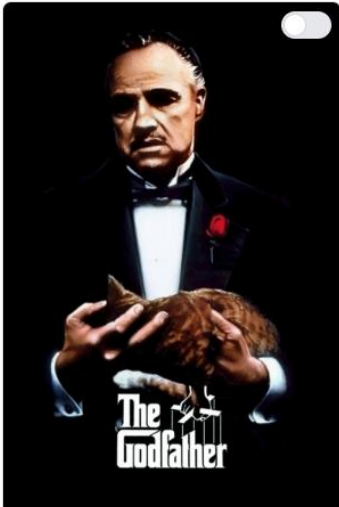
< > /movies

Movies

Grid View Table View



**The Shawshank Redemption**  
★ 8.7 • 1994  
Imprisoned in the 1940s for the double murder of his wife and her

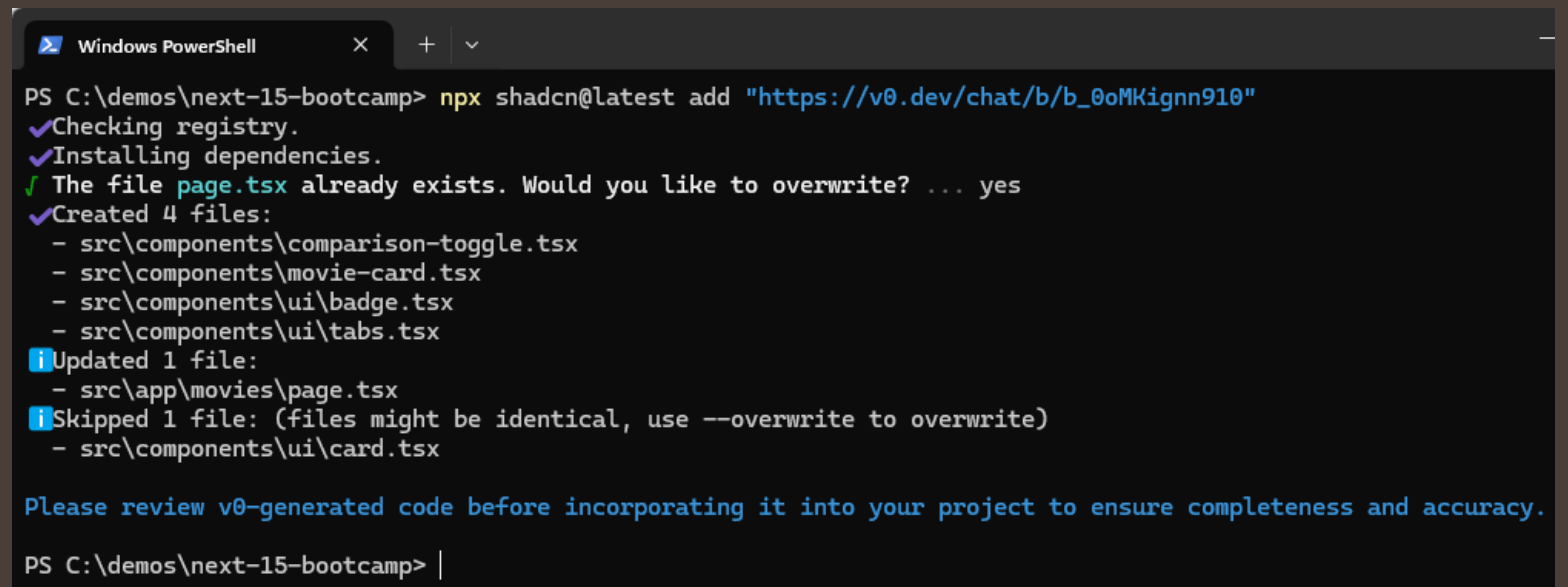


**The Godfather**  
★ 8.7 • 1972  
Spanning the years 1945 to 1955, a chronicle of the fictional Italian-American Corleone crime family...

Console

# Rendering movie cards

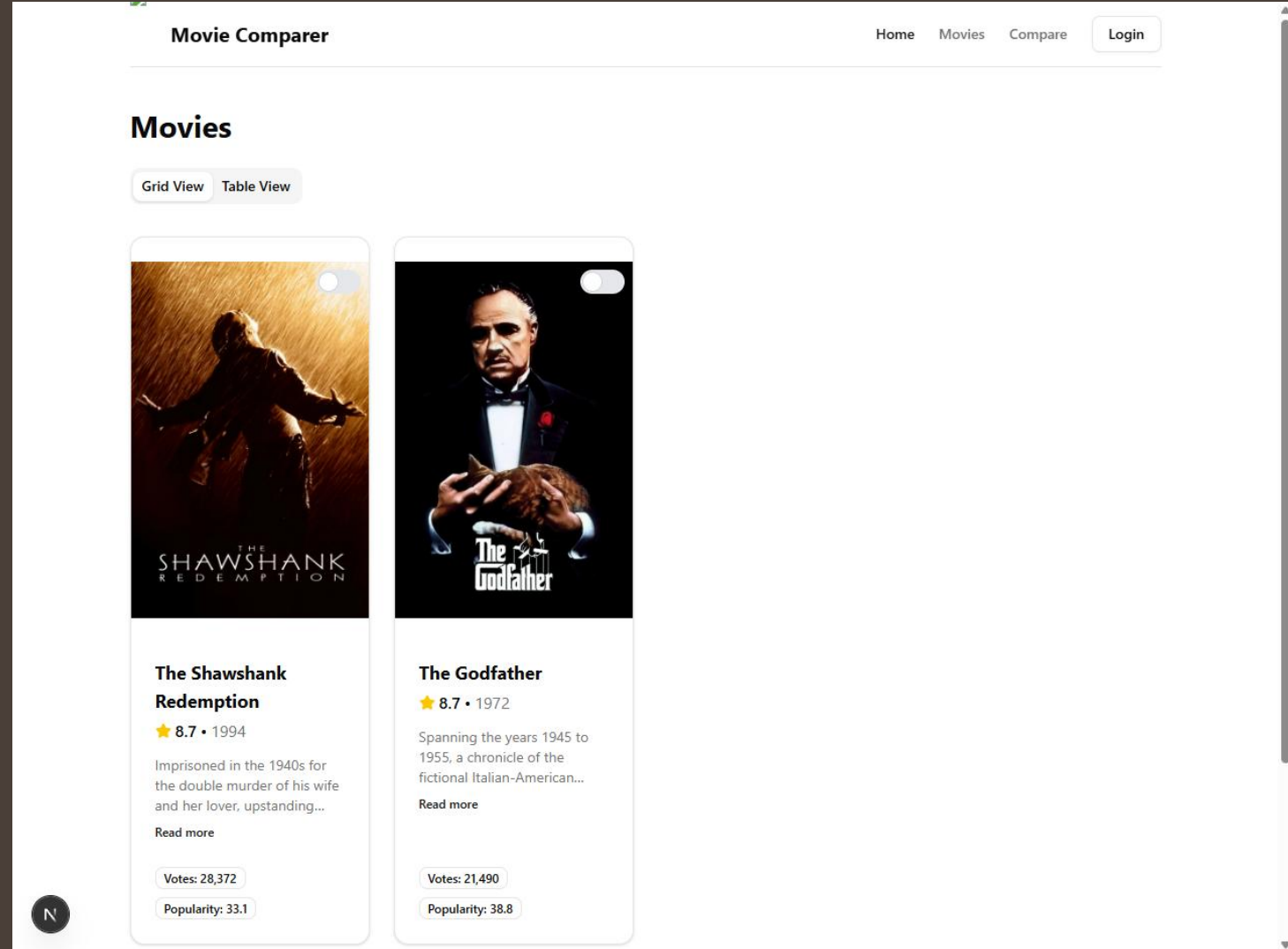
- `npx shadcn@latest add "https://vo.dev/chat/b/b_uCxqguwB2Qu"`



```
Windows PowerShell
PS C:\demos\next-15-bootcamp> npx shadcn@latest add "https://v0.dev/chat/b/b_0oMKiggn910"
✓Checking registry.
✓Installing dependencies.
✓ The file page.tsx already exists. Would you like to overwrite? ... yes
✓Created 4 files:
- src\components\comparison-toggle.tsx
- src\components\movie-card.tsx
- src\components\ui\badge.tsx
- src\components\ui\tabs.tsx
iUpdated 1 file:
- src\app\movies\page.tsx
iSkipped 1 file: (files might be identical, use --overwrite to overwrite)
- src\components\ui\card.tsx

Please review v0-generated code before incorporating it into your project to ensure completeness and accuracy.
PS C:\demos\next-15-bootcamp> |
```

# Rendering movie cards

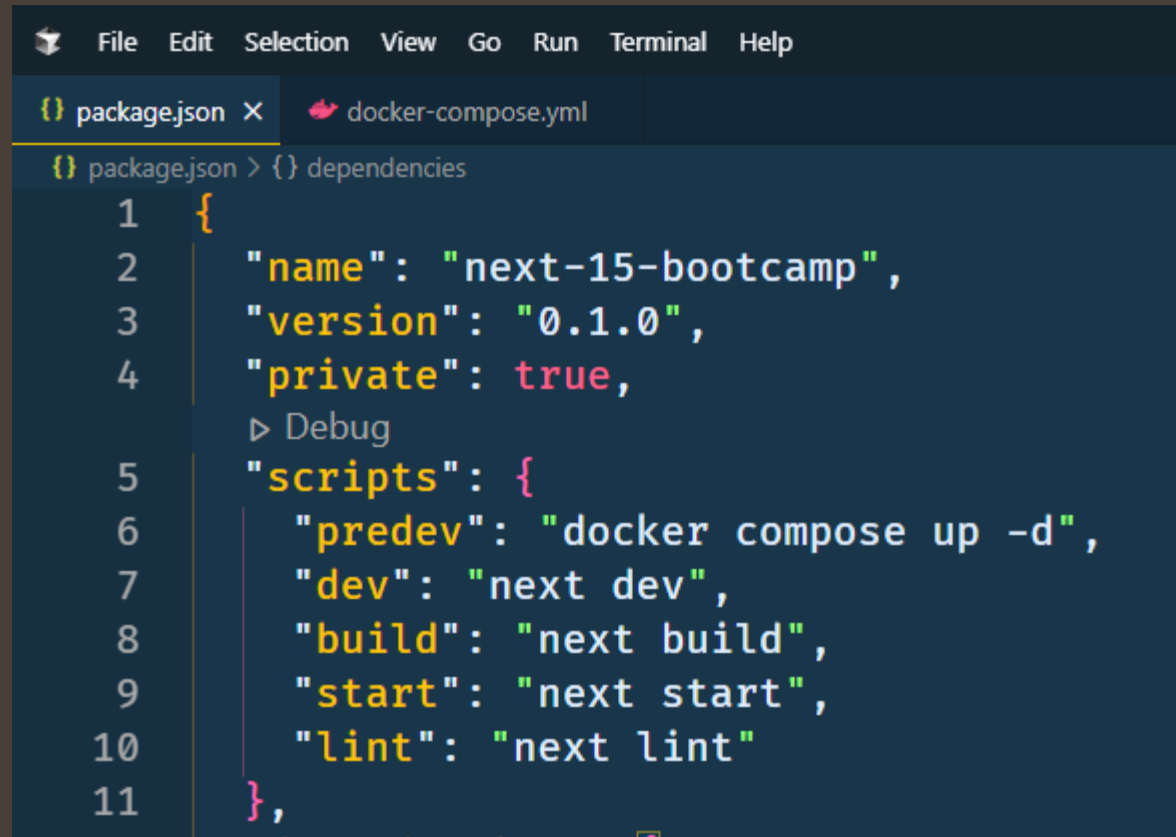


# Adding a Postgres SQL database using Docker

# Why use Docker?

- **Environment consistency**
  - Every developer on your team gets exactly the same database version, configuration, and extensions, eliminating "works on my machine" issues that plague traditional local installations.
- **Zero installation complexity**
  - Skip the headache of installing Postgres natively, managing system services, or dealing with permission issues. One docker run command gets you a fully functional database.
- **Isolated and disposable**
  - Each project can have its own containerized database that won't interfere with other projects or your system. When you're done, simply delete the container with no leftover files or configurations.
- **Resource control**
  - Easily limit memory and CPU usage for your development database, preventing it from consuming too many system resources during intensive operations.
- **Quick reset capabilities**
  - Corrupted data or need a fresh start? Destroy and recreate your database container in seconds rather than manually dropping/recreating schemas or reinstalling software.

# Adding a Postgres SQL database



```
File Edit Selection View Go Run Terminal Help
package.json X docker-compose.yml
package.json > {} dependencies
1  {
2    "name": "next-15-bootcamp",
3    "version": "0.1.0",
4    "private": true,
5    "scripts": {
6      "predev": "docker compose up -d",
7      "dev": "next dev",
8      "build": "next build",
9      "start": "next start",
10     "lint": "next lint"
11   },
```

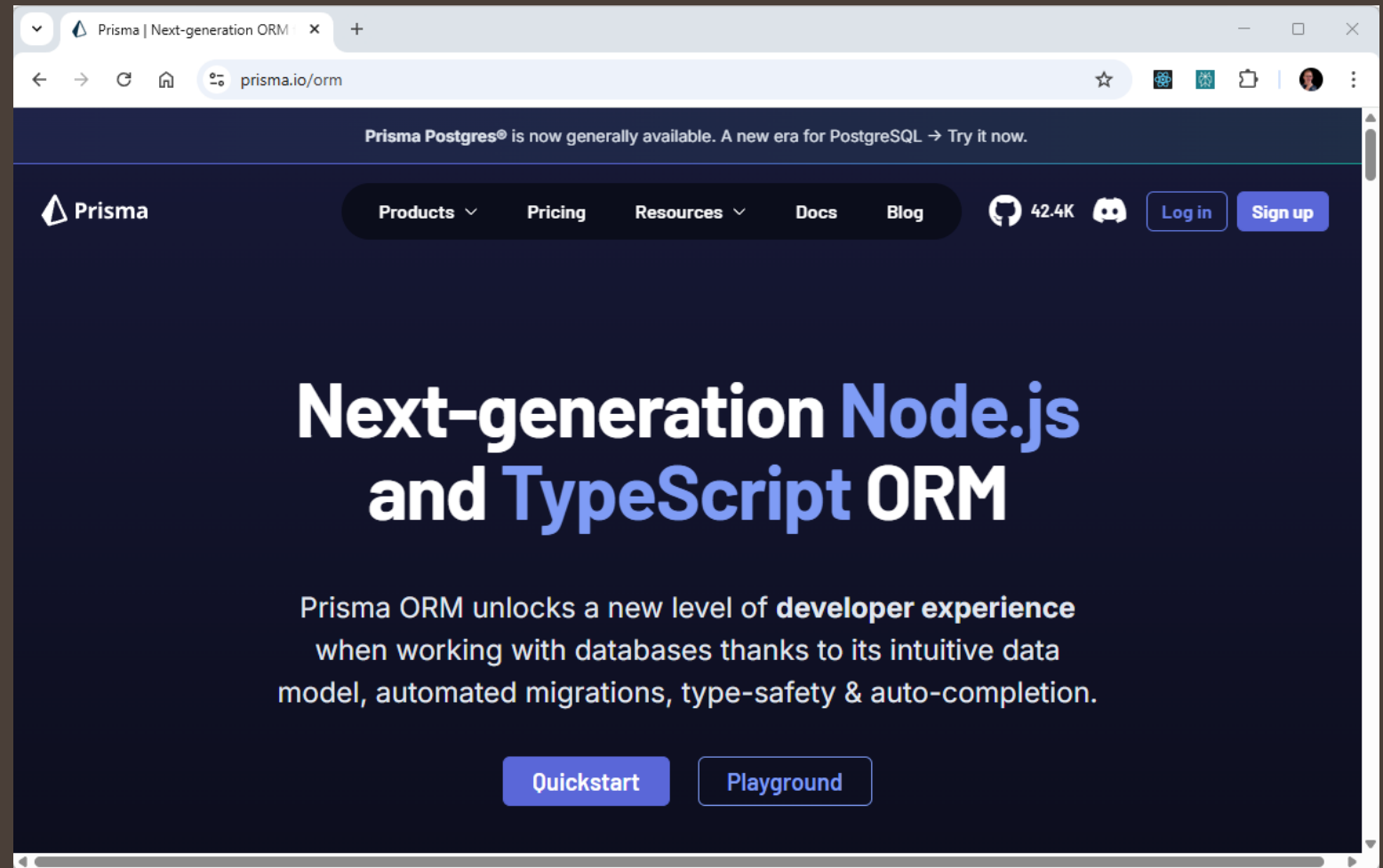
# Adding a Postgres SQL database



```
File Edit Selection View Go Run Terminal Help
package.json docker-compose.yml X
docker-compose.yml
1  services:
2    postgres:
3      image: postgres:17-alpine
4      container_name: Movie-Comparer-Postgres
5      environment:
6        POSTGRES_DB: postgres
7        POSTGRES_USER: postgres
8        POSTGRES_PASSWORD: postgres
9      ports:
10       - '54323:5432'
11      volumes:
12       - postgres_data:/var/lib/postgresql/data
13
14 volumes:
15   postgres_data:
```

# Adding the Prisma ORM

# Adding the Prisma ORM



# Adding the Prisma ORM

- **Type Safety Throughout**
  - Prisma generates TypeScript types automatically from your database schema, ensuring complete type safety from database queries to your Next.js components, eliminating runtime errors from type mismatches.
- **Seamless API Route Integration**
  - Prisma's query syntax works perfectly in Next.js API routes, server components and server actions, providing clean database operations without complex SQL or connection management overhead.
- **Excellent Developer Experience**
  - Prisma Studio provides a visual database browser, while the Prisma CLI offers schema migration tools and database seeding capabilities that streamline development workflow.
- **Database Agnostic Flexibility**
  - Switch between PostgreSQL, MySQL, SQLite, MongoDB, and other databases without changing your application code, making it easy to adapt to different deployment environments.

# Adding the Prisma ORM

- **Optimized for Modern Next.js Features**
  - Works seamlessly with App Router, Server Components, and React Server Components, allowing you to fetch data directly in components without additional API layers.
- **Built-in Connection Pooling**
  - Handles database connections efficiently, particularly important in serverless Next.js deployments where connection management can be challenging.
- **Powerful Migration System**
  - Prisma Migrate tracks schema changes and generates migration files automatically, making database versioning and deployment straightforward across different environments.
- **Query Optimization and Caching**
  - Provides query result caching and includes tools to analyze and optimize database performance, crucial for Next.js applications that need fast response times.

# Adding the Prisma ORM

```
Windows PowerShell
PS C:\demos\next-15-bootcamp> npm install prisma --save-dev

added 7 packages, and audited 376 packages in 7s

138 packages are looking for funding
  run 'npm fund' for details

found 0 vulnerabilities
PS C:\demos\next-15-bootcamp> npx prisma init --datasource-provider postgresql --output ../generated/prisma
Fetching latest updates for this subcommand...

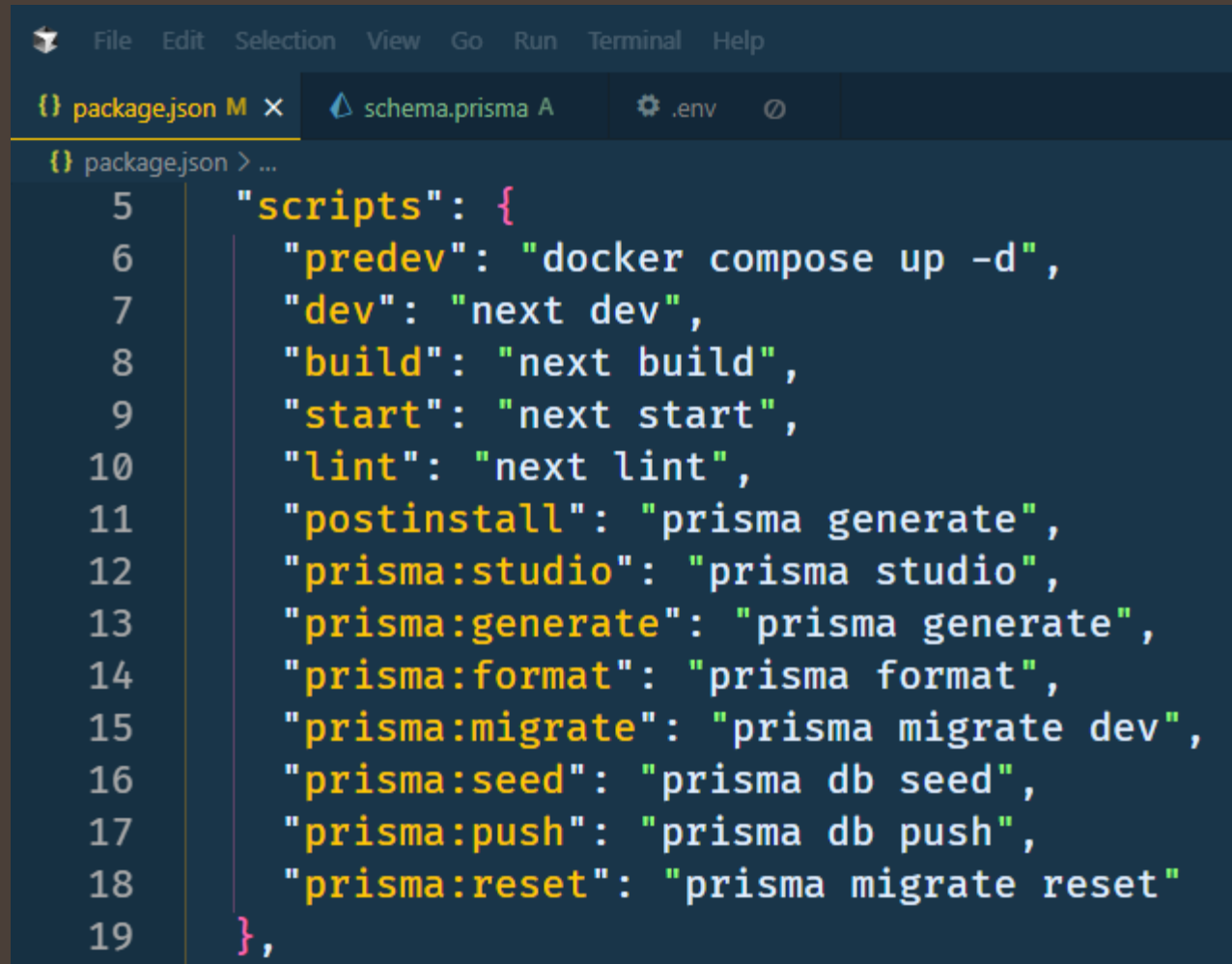
✓ Your Prisma schema was created at prisma/schema.prisma
  You can now open it in your favorite editor.

warn You already have a .gitignore file. Don't forget to add '.env' in it to not commit any private information.

Next steps:
1. Set the DATABASE_URL in the .env file to point to your existing database. If your database has no tables yet,
   read https://pris.ly/d/getting-started
2. Run prisma db pull to turn your database schema into a Prisma schema.
3. Run prisma generate to generate the Prisma Client. You can then start querying your database.
4. Tip: Explore how you can extend the ORM with scalable connection pooling, global caching, and real-time datab
   ase events. Read: https://pris.ly/cli/beyond-orm

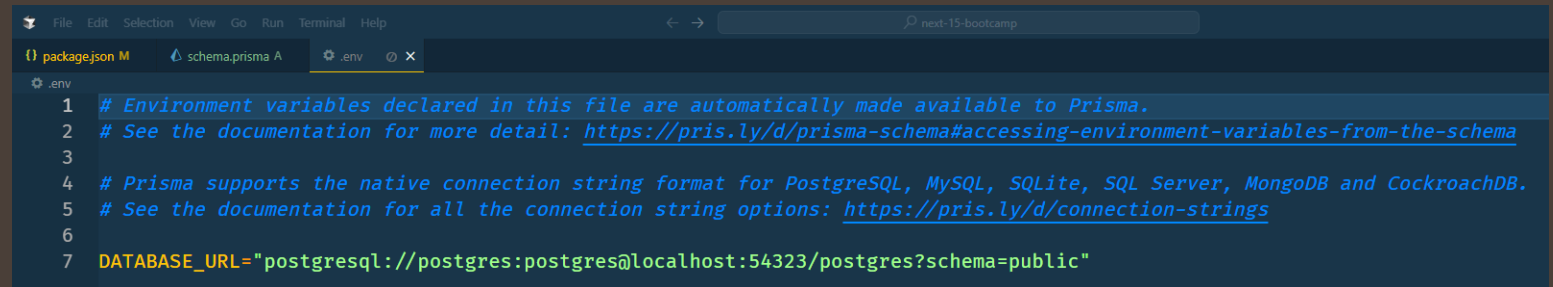
More information in our documentation:
https://pris.ly/d/getting-started
```

# Adding the Prisma ORM



```
File Edit Selection View Go Run Terminal Help
package.json M x schema.prisma A .env
package.json > ...
5  "scripts": {
6    "predev": "docker compose up -d",
7    "dev": "next dev",
8    "build": "next build",
9    "start": "next start",
10   "lint": "next lint",
11   "postinstall": "prisma generate",
12   "prisma:studio": "prisma studio",
13   "prisma:generate": "prisma generate",
14   "prisma:format": "prisma format",
15   "prisma:migrate": "prisma migrate dev",
16   "prisma:seed": "prisma db seed",
17   "prisma:push": "prisma db push",
18   "prisma:reset": "prisma migrate reset"
19 },
```

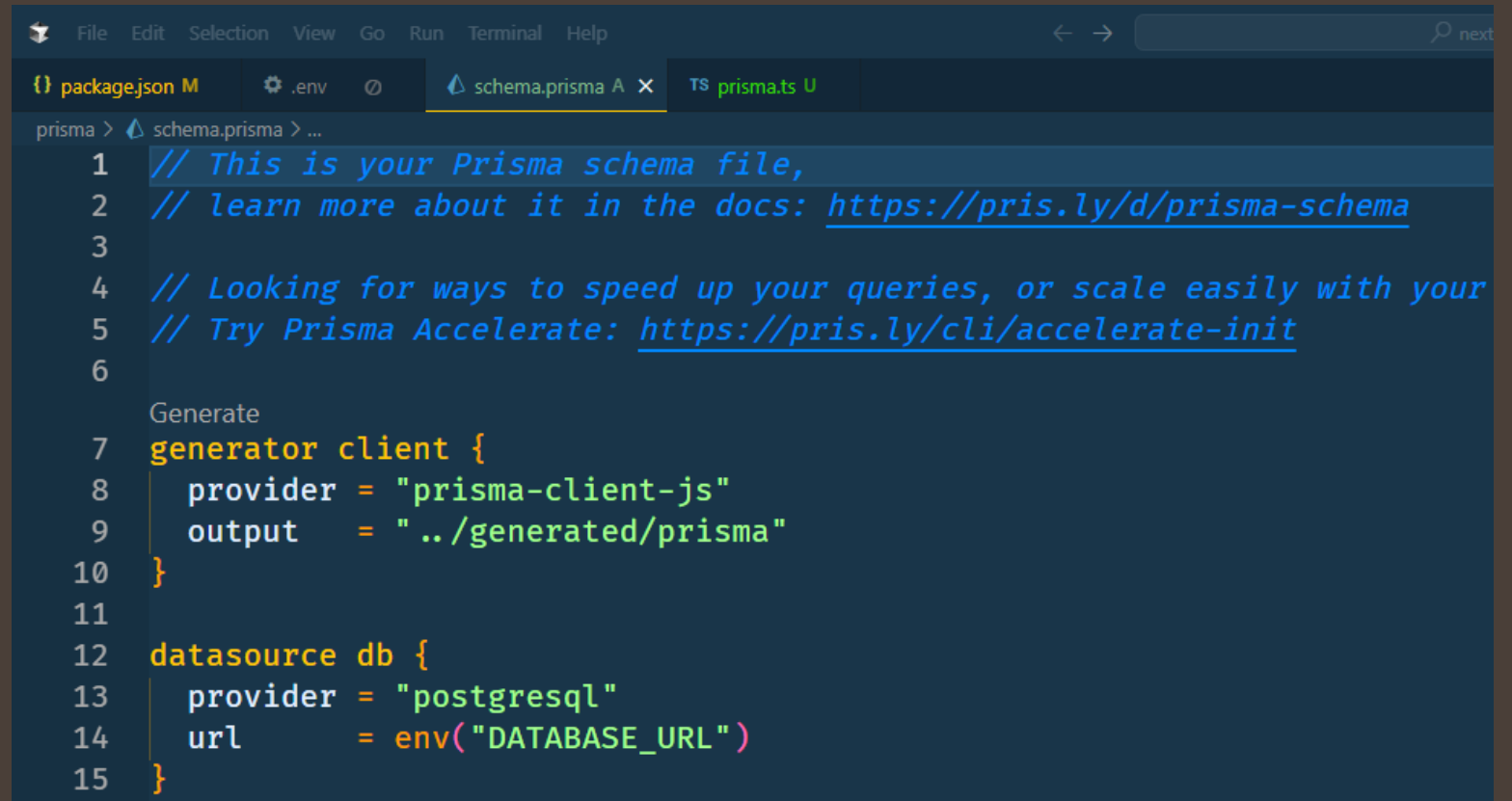
# Adding the Prisma ORM



The screenshot shows a code editor with a dark theme. The top bar includes a menu (File, Edit, Selection, View, Go, Run, Terminal, Help) and a search bar with the text 'next-15-bootcamp'. Below the menu, there are three tabs: 'package.json M', 'schema.prisma A', and '.env'. The '.env' tab is active, showing the following content:

```
.env
1 # Environment variables declared in this file are automatically made available to Prisma.
2 # See the documentation for more detail: https://pris.ly/d/prisma-schema#accessing-environment-variables-from-the-schema
3
4 # Prisma supports the native connection string format for PostgreSQL, MySQL, SQLite, SQL Server, MongoDB and CockroachDB.
5 # See the documentation for all the connection string options: https://pris.ly/d/connection-strings
6
7 DATABASE_URL="postgresql://postgres:postgres@localhost:54323/postgres?schema=public"
```

# Adding the Prisma ORM



```
prisma > schema.prisma > ...
1 // This is your Prisma schema file,
2 // learn more about it in the docs: https://pris.ly/d/prisma-schema
3
4 // Looking for ways to speed up your queries, or scale easily with your
5 // Try Prisma Accelerate: https://pris.ly/cli/accelerate-init
6
7 Generate
8 generator client {
9   provider = "prisma-client-js"
10  output   = "../generated/prisma"
11 }
12
13 datasource db {
14   provider = "postgresql"
15   url      = env("DATABASE_URL")
16 }
```

# Adding the Prisma ORM

```
Windows PowerShell X Windows PowerShell X + v
PS C:\demos\next-15-bootcamp> npx prisma migrate dev --name init
Environment variables loaded from .env
Prisma schema loaded from prisma\schema.prisma
Datasource "db": PostgreSQL database "postgres", schema "public" at "localhost:54323"

Already in sync, no schema change or pending migration was found.


✔Generated Prisma Client (v6.8.2) to .\generated\prisma in 32ms

PS C:\demos\next-15-bootcamp> |
```


# Adding the Prisma ORM (Next.js)



```
File Edit Selection View Go Run Terminal Help
package.json M .env schema.prisma A TS prisma.ts U x
src > lib > TS prisma.ts > ...
1 import { PrismaClient } from '../generated/prisma';
2
3 const globalForPrisma = global as unknown as {
4   prisma: PrismaClient;
5 };
6
7 const prisma = globalForPrisma.prisma || new PrismaClient();
8
9 if (process.env.NODE_ENV !== 'production') globalForPrisma.prisma = prisma;
10
11 export default prisma;
```



# Adding the database schema



## Adding the database schema

- Prisma database schemas
  - Prisma uses a `schema.prisma` file as a single source of truth that defines your database structure, models, and relationships in a declarative format.
  - This schema serves as the foundation for Prisma's code generation, allowing it to automatically create type-safe client code that matches your exact database structure.
- Migrations are created in development to do schema updates
  - `prisma migrate dev --name <migration name>`
- Migrations are executed in production to update the production DB
  - `prisma migrate deploy`

# Adding the database schema

```
File Edit Selection View Go Run Terminal Help
schema.prisma M x
prisma > schema.prisma > ...
17 model Movie {
18   id      Int      @id @default(autoincrement())
19   title    String
20   overview String
21   genres   Genre[]
22   releaseDate String
23   backdropPath String
24   posterPath String
25   popularity Float
26   voteAverage Float
27   voteCount Int
28   actors   Actor[]
29   directors Director[]
30   favourite Boolean @default(false)
31 }
32
33 model Genre {
34   id      Int      @id @default(autoincrement())
35   name     String
36   movies  Movie[]
37 }
38
39 model Actor {
40   id      Int      @id @default(autoincrement())
41   name     String
42   movies  Movie[]
43 }
44
45 model Director {
46   id      Int      @id @default(autoincrement())
47   name     String
48   movies  Movie[]
```

## Adding the database schema



```
Windows PowerShell X Windows PowerShell X + v
PS C:\demos\next-15-bootcamp> npx prisma migrate dev --name movies
Environment variables loaded from .env
Prisma schema loaded from prisma\schema.prisma
Datasource "db": PostgreSQL database "postgres", schema "public" at "localhost:54323"

Applying migration `20250602192938_movies`

The following migration(s) have been created and applied from new schema changes:

migrations/
├─ 20250602192938_movies/
└─ migration.sql

Your database is now in sync with your schema.

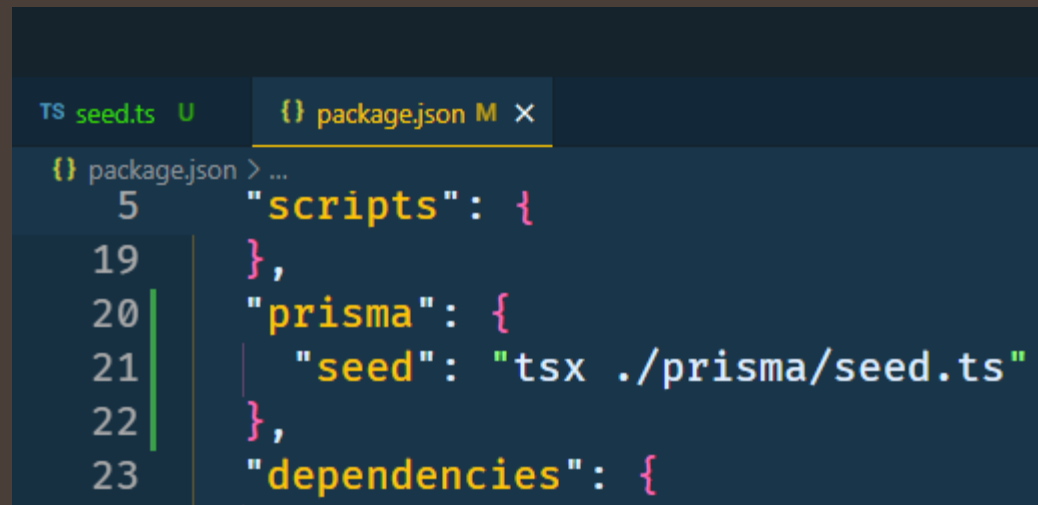
✓Generated Prisma Client (v6.8.2) to .\generated\prisma in 79ms
```

# Seeding the database

# Seeding the database

- **Consistent Development Environment**
  - Database seeding ensures all developers on your team start with the same baseline data, eliminating inconsistencies that can cause bugs to appear on some machines but not others and making debugging more predictable across different development setups.
- **Faster Feature Development and Testing**
  - Having realistic sample data immediately available allows developers to test user interfaces, business logic, and edge cases without manually creating test records every time they reset their local database or onboard new team members.
- **Automated Deployment Pipeline Integration**
  - Seed scripts can be integrated into your CI/CD pipeline to populate staging and testing environments with known data sets, enabling automated testing scenarios and providing consistent environments for QA teams to validate features.
- **Realistic Data Relationships and Volume**
  - Well-designed seed data includes proper foreign key relationships, realistic data volumes, and edge cases that mirror production scenarios, helping identify performance issues and relationship problems early in development rather than after deployment.

# Seeding the database



The screenshot shows a code editor with two tabs: 'TS seed.ts U' and '{} package.json M X'. The 'package.json' tab is active, displaying the following JSON structure:

```
{  
  "scripts": {  
    "prisma": {  
      "seed": "tsx ./prisma/seed.ts"  
    },  
    "dependencies": {  

```

The line numbers 5, 19, 20, 21, 22, and 23 are visible on the left side of the editor, corresponding to the lines of code.

# Seeding the database

```
TS seed.ts U x {} packagejson M
prisma > TS seed.ts > ...
1 import movies from './seed-data/movies.json';
2 import genres from './seed-data/genres.json';
3 import actors from './seed-data/actors.json';
4 import directors from './seed-data/directors.json';
5 import { PrismaClient } from '../generated/prisma';
6
7 const prisma = new PrismaClient();
8
9 async function main() {
10   for (const actor of actors) {
11     await prisma.actor.upsert({
12       where: { id: actor.id },
13       update: actor,
14       create: actor,
15     });
16   }
17
18   for (const director of directors) { ...
24   }
25
26   for (const genre of genres) { ...
32   }
33
34   for (const item of movies) {
35     const { genreIds, actorIds, directorIds, ...rest } = item;
36     const movie = {
37       ...rest,
38       favourite: false,
39       genres: {
40         connect: genreIds.map((id) => ({ id })),
41       },
42       actors: {
43         connect: actorIds.map((id) => ({ id })),
44       },

```

# Seeding the database



```
Windows PowerShell X Windows PowerShell X
PS C:\demos\next-15-bootcamp> npm run prisma:seed

> next-15-bootcamp@0.1.0 prisma:seed
> prisma db seed

Environment variables loaded from .env
Running seed command `tsx ./prisma/seed.ts` ...

The seed command has been executed.
PS C:\demos\next-15-bootcamp> |
```



Using data from the DB

# Using data from the DB

- **Direct Database Queries in Components**
  - Server components allow you to write Prisma database queries directly inside React components that render on the server, eliminating the need for separate API routes or data fetching layers for many common use cases like displaying lists or detail pages.
- **Async Component Rendering**
  - React 19 server components can be async functions, meaning you can use `await` directly in component code to fetch data from Prisma before rendering, making data fetching feel more natural and reducing the complexity of managing loading states for initial page loads.
- **Improved Performance and SEO**
  - Since Prisma queries execute on the server and components render with data already available, pages load faster with complete HTML content, improving SEO and eliminating the "loading spinner then content" pattern that hurts user experience and search rankings.
- **Reduced Client-Server Roundtrips**
  - Server components with embedded Prisma queries eliminate the need for separate API calls after page load, reducing network requests and improving perceived performance, especially for data that doesn't need to be dynamically updated on the client.
- **Simplified Error Handling and Security**
  - Database connection strings and Prisma client configuration remain server-side only, improving security while allowing you to handle database errors closer to where queries are executed rather than managing error states across API boundaries.

# Server vs Client Components

- **Server Components by Default**
  - In React 19 with Next.js app router, components are server components by default, meaning they render on the server during build time or request time, have access to server-side resources like databases and file systems, and cannot use browser-specific APIs or event handlers.
- **Client Component Declaration**
  - To create a client component that runs in the browser, you must add the 'use client' directive at the very top of your component file before any imports, which tells React to hydrate and run this component on the client side with full interactivity.
- **Interactivity and State Management**
  - Client components can use React hooks like `useState`, `useEffect`, and event handlers like `onClick`, while server components cannot use any interactive features, browser APIs, or state management since they only exist during server rendering.
- **Bundle Size and Performance Trade-offs**
  - Server components don't add to your JavaScript bundle size since they render on the server, while client components increase bundle size and require hydration, so you should only mark components as client components when interactivity is actually needed.
- **Data Fetching Patterns**
  - Server components can directly access databases, APIs, and server resources during rendering, while client components must fetch data through client-side methods like `fetch()` calls to API routes or external endpoints after the component mounts.
- **Component Composition Rules**
  - Server components can import and render client components, but client components cannot directly import server components, though server components can be passed as props or children to client components for flexible architectures.

# Using data from the DB



```
File Edit Selection View Go Run Terminal Help
page.tsx 3, M x
src > app > movies > page.tsx > ...
1 import { MovieCard } from '@components/movie-card';
2 import { Tabs, TabsContent, TabsList, TabsTrigger } from '@components/ui/tabs';
3 import { ComparisonToggle } from '@components/comparison-toggle';
4 import prisma from '@lib/prisma';
5
6 export default async function MoviesPage() {
7   const movies = await prisma.movie.findMany({
8     orderBy: {
9       releaseDate: 'desc',
10     },
11   });
12
13   return (
14     <div className="container py-10">
15       <h1 className="text-3xl font-bold mb-6">Movies</h1>
```

# Adding client-side interactions and state

Using [Valtio](#) - Proxy state made simple 😊

# Adding client-side interactions and state

- Proxy-Based Reactivity
  - [Valtio](#) uses JavaScript proxies to automatically track state mutations and re-render components only when the specific data they access changes, eliminating the need for manual dependency arrays or complex state update patterns that can cause unnecessary re-renders.
- Mutable State Updates
  - Unlike Redux or Zustand, Valtio allows you to directly mutate state objects using familiar JavaScript syntax like `state.user.name = 'John'`, making state updates feel natural and reducing boilerplate code while still maintaining React's reactivity principles.
- Minimal Boilerplate and Learning Curve
  - Valtio requires very little setup code and uses intuitive APIs that feel like working with plain JavaScript objects, making it easier for developers to adopt compared to more complex state management solutions that require actions, reducers, or special update functions.
- Automatic Optimization
  - The library automatically optimizes re-renders by tracking which properties each component accesses, so components only update when their specific data dependencies change, leading to better performance without manual optimization efforts like memoization or selector functions.
- TypeScript Integration
  - Valtio provides excellent TypeScript support with full type inference for state objects and mutations, ensuring type safety across your application while maintaining the simple mutable API that makes the library appealing.
- Flexible State Architecture
  - You can organize state into multiple proxy objects for different domains of your application, share state across components easily, and even subscribe to state changes outside of React components for integration with other libraries or side effects.

# Adding client-side interactions and state

```
File Edit Selection View Go Run Terminal Help
TS store.ts A X comparison-toggle.tsx M movie-card.tsx 1, M page.tsx 1, M
src > lib > TS store.ts > ...
1 import { proxy } from 'valtio';
2 import { Prisma } from '../generated/prisma';
3
4 export type MovieToStore = Prisma.MovieGetPayload<{
5   include: {
6     actors: true;
7     directors: true;
8     genres: true;
9   };
10 }>;
11
12 interface MovieStore {
13   comparingMovies: MovieToStore[];
14 }
15
16 const initialState: MovieStore = {
17   comparingMovies: [],
18 };
19
20 export const movieStore = proxy<MovieStore>(initialState);
21
22 export const movieActions = {
23   toggleCompare: (movie: MovieToStore) => {
24     if (!movieStore.comparingMovies.map((m) => m.id).includes(movie.id)) {
25       movieStore.comparingMovies = [...movieStore.comparingMovies, movie];
26     } else {
27       movieStore.comparingMovies = movieStore.comparingMovies.filter(
28         (m) => m.id !== movie.id
29       );
30     }
31   },
32 };
33
```

# Adding client-side interactions and state

```
File Edit Selection View Go Run Terminal Help
TS store.ts U comparison-toggle.tsx M X movie-card.tsx 1, M page.tsx 1, M
src > components > comparison-toggle.tsx > ...
1 | 'use client';
2 |
3 | import { useSnapshot } from 'valtio';
4 | import { movieActions, movieStore, MovieToStore } from '@lib/store';
5 |
6 | interface ComparisonToggleProps {
7 |   movie: MovieToStore;
8 | }
9 |
10 | export function ComparisonToggle({ movie }: ComparisonToggleProps) {
11 |   const snap = useSnapshot(movieStore);
12 |   const isComparing = !!snap.comparingMovies.find((m) => m.id === movie.id);
13 |
14 |   return (
15 |     <label className="inline-flex items-center cursor-pointer">
16 |       <input
17 |         type="checkbox"
18 |         className="sr-only peer"
19 |         checked={isComparing}
20 |         onChange={() => movieActions.toggleCompare(movie)}
21 |       />
22 |       <div className="relative w-11 h-6 bg-gray-200 peer-focus:outline-none"
23 |         <span className="sr-only">Compare</span>
24 |     </div>
25 |   );
26 | }
27 |
```

# Adding client-side interactions and state

```
File Edit Selection View Go Run Terminal Help
TS store.ts U comparison-toggle.tsx M movie-card.tsx 1, M X page.tsx 1, M
src > components > movie-card.tsx > ...
1 'use client';
2
3 import { Card, CardContent, CardFooter } from '@components/ui/card';
4 import { Badge } from '@components/ui/badge';
5 import { Star } from 'lucide-react';
6 import { useState } from 'react';
7 import { ComparisonToggle } from '@components/comparison-toggle';
8 import { MovieToStore } from '@lib/store';
9
10 interface MovieCardProps {
11   movie: MovieToStore;
12 }
13
14 export function MovieCard({ movie }: MovieCardProps) {
15   const [isExpanded, setIsExpanded] = useState(false);
16
17   return (
18     <Card className="overflow-hidden h-full flex flex-col">
19       <div className="relative">
20         <img
21           src={movie.posterPath || '/placeholder.svg'}
22           alt={movie.title}
23           className="w-full aspect-[2/3] object-cover"
24         />
25         <div className="absolute top-2 right-2">
26           <ComparisonToggle movie={movie} />
27         </div>
28       </div>
29       <CardContent className="pt-4 flex-grow">
```

# Adding client-side interactions are



```
File Edit Selection View Go Run Terminal Help
src > app > movies > page.tsx > ...
1 import { MovieCard } from '@components/movie-card';
2 import { Tabs, TabsContent, TabsList, TabsTrigger } from '@components/ui/tabs';
3 import { ComparisonToggle } from '@components/comparison-toggle';
4 import prisma from '@lib/prisma';
5
6 export default async function MoviesPage() {
7   const movies = await prisma.movie.findMany({
8     include: {
9       actors: true,
10       directors: true,
11       genres: true,
12     },
13     orderBy: {
14       releaseDate: 'desc',
15     },
16   });
17
18   return (
19     <div className="container py-10">
20       <h1 className="text-3xl font-bold mb-6">Movies</h1>
21
22       <Tabs defaultValue="grid" className="mb-8">
23         <div className="flex items-center justify-between">
24           <TabsList>
25             <TabsTrigger value="grid">Grid View</TabsTrigger>
26             <TabsTrigger value="table">Table View</TabsTrigger>
27           </TabsList>
28         </div>
29
30         <TabsContent value="grid" className="mt-6">
31           <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6">
32             {movies.map((movie) => (
33               <MovieCard key={movie.id} movie={movie} />
34             ))}
35           </div>
36         </TabsContent>

```



# Comparing movies

# Comparing movies

- **We want the user of the app to be able to compare movies**
  - We will use Vercel V0 again to generate the page
  - The generated components will use generated data
  - We will need to hook it up with the Valtio store

# Comparing movies

```
File Edit Selection View Go Run Terminal Help
3-movie-comparison.md x
prompts > 3-movie-comparison.md
1 I would like a page to display a number of movies side by side to compare them and decide which I
  want to watch. Show as much data about the movie as possible.
2
3 The route is /compare
4
5 The Prisma schema for the data looks like this:
6
7 ```
8 // This is your Prisma schema file,
9 // learn more about it in the docs: https://pris.ly/d/prisma-schema
10
11 // Looking for ways to speed up your queries, or scale easily with your serverless or edge functions?
12 // Try Prisma Accelerate: https://pris.ly/cli/accelerate-init
13
14 Generate
15 generator client {
16   provider = "prisma-client-js"
17   output   = "../generated/prisma"
18 }
19
20 datasource db {
21   provider = "postgresql"
22   url      = env("DATABASE_URL")
23 }
24
25 model Movie {
26   id          Int      @id @default(autoincrement())
27   title       String
28   overview    String
29   genres      Genre[]
30   releaseDate String
31   backdropPath String
32   posterPath  String
33   popularity  Float
34   voteAverage Float
35   voteCount   Int
36   actors      Actor[]
37   directors   Director[]
```

# Comparing movies

- `npx shadcn@latest add "https://v0.dev/chat/b/b_jwu1vRBKFBo"`

```
Windows PowerShell
PS C:\demos\next-15-bootcamp> npx shadcn@latest add "https://v0.dev/chat/b/b_jwu1vRBKFBo"
✓Checking registry.
✓Installing dependencies.
✓ The file page.tsx already exists. Would you like to overwrite? ... yes
✓ The file movie-comparison-card.tsx already exists. Would you like to overwrite? ... yes
i Updated 2 files:
- src\app\compare\page.tsx
- src\components\movie-comparison-card.tsx
i Skipped 4 files: (files might be identical, use --overwrite to overwrite)
- src\components\ui\card.tsx
- src\components\ui\badge.tsx
- src\components\ui\switch.tsx
- src\components\ui\label.tsx

Please review v0-generated code before incorporating it into your project to ensure completeness and accuracy.

PS C:\demos\next-15-bootcamp> |
```

# Comparing movies

```
File Edit Selection View Go Run Terminal Help
page.tsx M x movie-comparison-card.tsx M TS store.ts M
src > app > compare > page.tsx > ...
1 | 'use client';
2 |
3 | import { MovieComparisonCard } from '@components/movie-comparison-card';
4 | import { movieStore } from '@lib/store';
5 | import { useSnapshot } from 'valtio';
6 |
7 | export default function ComparePage() {
8 |   const snap = useSnapshot(movieStore);
9 |
10 |   return (
11 |     <div className="container mx-auto px-4 py-8">
12 |       <div className="mb-8">
13 |         <h1 className="text-4xl font-bold text-center mb-4">
14 |           Movie Comparison
15 |         </h1>
16 |         <p className="text-lg text-muted-foreground text-center max-w-2xl mx-auto">
17 |           Compare movies side by side to help you decide what to watch next.
18 |           Toggle off movies you're not interested in.
19 |         </p>
20 |       </div>
21 |
22 |       <div className="grid grid-cols-1 md:grid-cols-2 xl:grid-cols-3 2xl:grid-cols-4 gap-6">
23 |         {snap.comparingMovies.map((movie) => (
24 |           <MovieComparisonCard key={movie.id} movie={movie} />
25 |         ))}
26 |       </div>
27 |     </div>
28 |   );
29 | }
```

# Comparing movies

```
File Edit Selection View Go Run Terminal Help
page.tsx M movie-comparison-card.tsx M TS store.ts M
src > components > movie-comparison-card.tsx > ...
14 export function MovieComparisonCard({ movie }: MovieComparisonCardProps) {
15   const snap = useSnapshot(movieStore);
16   const isComparing = !!snap.comparingMovies.find((m) => m.id === movie.id);
17
18   const formatDate = (dateString: string) => { ...
24   };
25
26   const formatPopularity = (popularity: number) => {
27     return popularity.toFixed(1);
28   };
29
30   return (
31     <Card className="w-full max-w-sm mx-auto h-fit">
32       <CardHeader className="pb-4">
33         <div className="flex items-center justify-between mb-4">
34           <div className="flex items-center space-x-2">
35             <Switch
36               id={`movie-${movie.id}`}
37               checked={isComparing}
38               onChange={() => movieActions.toggleCompare(movie)}
39             />
40             <Label
41               htmlFor={`movie-${movie.id}`}
42               className="text-sm font-medium"
43             >
44               Consider
45             </Label>
46           </div>
47           {movie.favourite && (
48             <Heart className="h-5 w-5 fill-red-500 text-red-500" />
49           )}
50         </div>
```

# Comparing movies



```
File Edit Selection View Go Run Terminal Help
next-15-bootcamp
page.tsx M movie-comparison-card.tsx M store.ts M X
src > lib > TS store.ts > ...
1 | import { proxy, Snapshot } from 'valtio';
2 | import { Prisma } from '../generated/prisma';
3 |
4 | > export type MovieToStore = Prisma.MovieGetPayload<{ ...
10 | > };
11 |
12 | interface MovieStore {
13 |   comparingMovies: MovieToStore[];
14 | }
15 |
16 | const initialState: MovieStore = {
17 |   comparingMovies: [],
18 | };
19 |
20 | export const movieStore = proxy<MovieStore>(initialState);
21 |
22 | export const movieActions = {
23 |   toggleCompare: (movie: MovieToStore | Snapshot<MovieToStore>) => {
24 |     if (!movieStore.comparingMovies.map((m) => m.id).includes(movie.id)) {
25 |       movieStore.comparingMovies = [
26 |         ...movieStore.comparingMovies,
27 |         movie as MovieToStore,
28 |       ];
29 |     } else {
30 |       movieStore.comparingMovies = movieStore.comparingMovies.filter(
31 |         (m) => m.id !== movie.id
32 |       );
33 |     }
34 |   },
35 | };
36 |
```

Problems Output Debug Console Terminal Ports GitLens Azure Playwright Comments

```
PS C:\demos\next-15-bootcamp> npx tsc
PS C:\demos\next-15-bootcamp> npx tsc
PS C:\demos\next-15-bootcamp> npx tsc
```

Source Control

Comparing movies

Commit

Changes

- TS next.config.ts M
- page.tsx src/app/compare M
- movie-comparison-card.tsx src/components M
- TS store.ts src/lib M

Source Control Graph

GitLens commits: main

- Up to date with origin on GitHub 30 minutes ago
- Compare main with origin/main
  - on main • fetched 30 minutes ago
- origin Adding client-side interactions and state You...
- Comparing movies from V0.dev You, 8 hours ago
- Rendering movie cards You, 3 days ago
- Seeding the database You, 4 days ago
- Adding the database schema You, 4 days ago
- Adding the Prisma ORM You, 4 days ago
- Adding a Postgres SQL database You, 5 days ago
- Using shared layouts You, 5 days ago
- V0 generated home page You, 5 days ago
- Over a week ago
- Initial commit from Create Next App You, last week

Cursor Tab Zen Ln 1, Col 1 Spaces: 2 UTF-8 CRLF TypeScript Prettier

Disabling the  
/compare route

# Disabling the /compare route

- **Prevent Empty State Confusion**
  - Disabling the compare route when no movies are selected prevents users from landing on a blank or confusing page that doesn't provide any value, eliminating the need to handle empty state UI and reducing user frustration with dead-end navigation.

# Disabling the /compare route

```
File Edit Selection View Go Run Terminal Help
next-15-bootcamp
header.tsx M page.tsx M
src > components > header.tsx > ...
1 | 'use client';
2 |
3 | import Link from 'next/link';
4 | import Image from 'next/image';
5 | import { Menu } from 'lucide-react';
6 | import { useSnapshot } from 'valtio';
7 | import { movieStore } from '@lib/store';
8 |
9 | import { Button } from '@components/ui/button';
10 | import { Sheet, SheetContent, SheetTrigger } from '@components/ui/sheet';
11 |
12 | export function Header() {
13 |   const snap = useSnapshot(movieStore);
14 |   const compareDisabled = snap.comparingMovies.length < 2;
15 |
16 |   return (
17 |     <header className="sticky top-0 z-40 w-full border-b bg-background">
18 |       <div className="container flex h-16 items-center justify-between">
19 |         <div className="flex items-center gap-2">...
22 |       </div>
23 |
24 |       { /* Desktop Navigation */ }
25 |       <nav className="hidden md:flex items-center gap-6">
26 |         <Link href="/" className="text-sm font-medium text-primary">...
28 |       </Link>
29 |         <Link href="/movies" className="text-sm font-medium text-muted-foreground hover:text-primary">...
31 |       </Link>
32 |         <Link
33 |           href="/compare"
34 |           className={`text-sm font-medium text-muted-foreground hover:text-primary ${
35 |             compareDisabled
36 |               ? 'opacity-50 pointer-events-none cursor-not-allowed'
37 |               : ''
38 |           }`}
39 |           aria-disabled={compareDisabled}
40 |           tabIndex={compareDisabled ? -1 : 0}
41 |         >
42 |         Compare
```

# Disabling the /compare route



```
File Edit Selection View Go Run Terminal Help
next-15-bootcamp
header.tsx M page.tsx M X
src > app > compare > page.tsx > ...
1 'use client';
2
3 import { MovieComparisonCard } from '@components/movie-comparison-card';
4 import { movieStore } from '@lib/store';
5 import { useSnapshot } from 'valtio';
6 import { useEffect } from 'react';
7 import { useRouter } from 'next/navigation';
8
9 export default function ComparePage() {
10   const snap = useSnapshot(movieStore);
11   const router = useRouter();
12
13   useEffect(() => {
14     if (snap.comparingMovies.length === 0) {
15       router.replace('/movies');
16     }
17   }, [snap.comparingMovies.length, router]);
18
19   return (
20     <div className="container mx-auto px-4 py-8">
21       <div className="mb-8">
22         <h1 className="text-4xl font-bold text-center mb-4">
23           Movie Comparison
24         </h1>
25         <p className="text-lg text-muted-foreground text-center max-w-2xl mx-auto">
26           Compare movies side by side to help you decide what to watch next.
27           Toggle off movies you're not interested in.
28         </p>
29       </div>
30
31       <div className="grid grid-cols-1 md:grid-cols-2 xl:grid-cols-3 2xl:grid-cols-4 gap-6">
32         {snap.comparingMovies.map((movie) => (
33           <MovieComparisonCard key={movie.id} movie={movie} />
34         ))}
35       </div>
36     </div>
37   );
38 }
```

# Deploying the database to the cloud

# Deploying the database to the cloud

- **Self-managed on Cloud VMs (DigitalOcean, Linode)**
  - Provides maximum control and potentially lower costs for high-usage scenarios, but requires significant database administration expertise for maintenance, security, and optimization tasks.
- **Managed Cloud Services (AWS, Google, Azure)**
  - Major cloud providers offer fully managed PostgreSQL with automated backups, patching, and scaling, but require more configuration and typically have higher costs with traditional instance-based pricing models.
- **Neon**
  - Provides serverless PostgreSQL with automatic scaling, database branching for development workflows, and a generous free tier, making it ideal for modern applications that need flexible compute resources without infrastructure management.
- **Supabase**
  - Combines PostgreSQL with a full backend-as-a-service platform including real-time subscriptions, authentication, and auto-generated APIs, perfect for developers who want a complete backend solution beyond just the database.
- **And many more**
  - Heroku, Aiven, Exoscale, etc ...

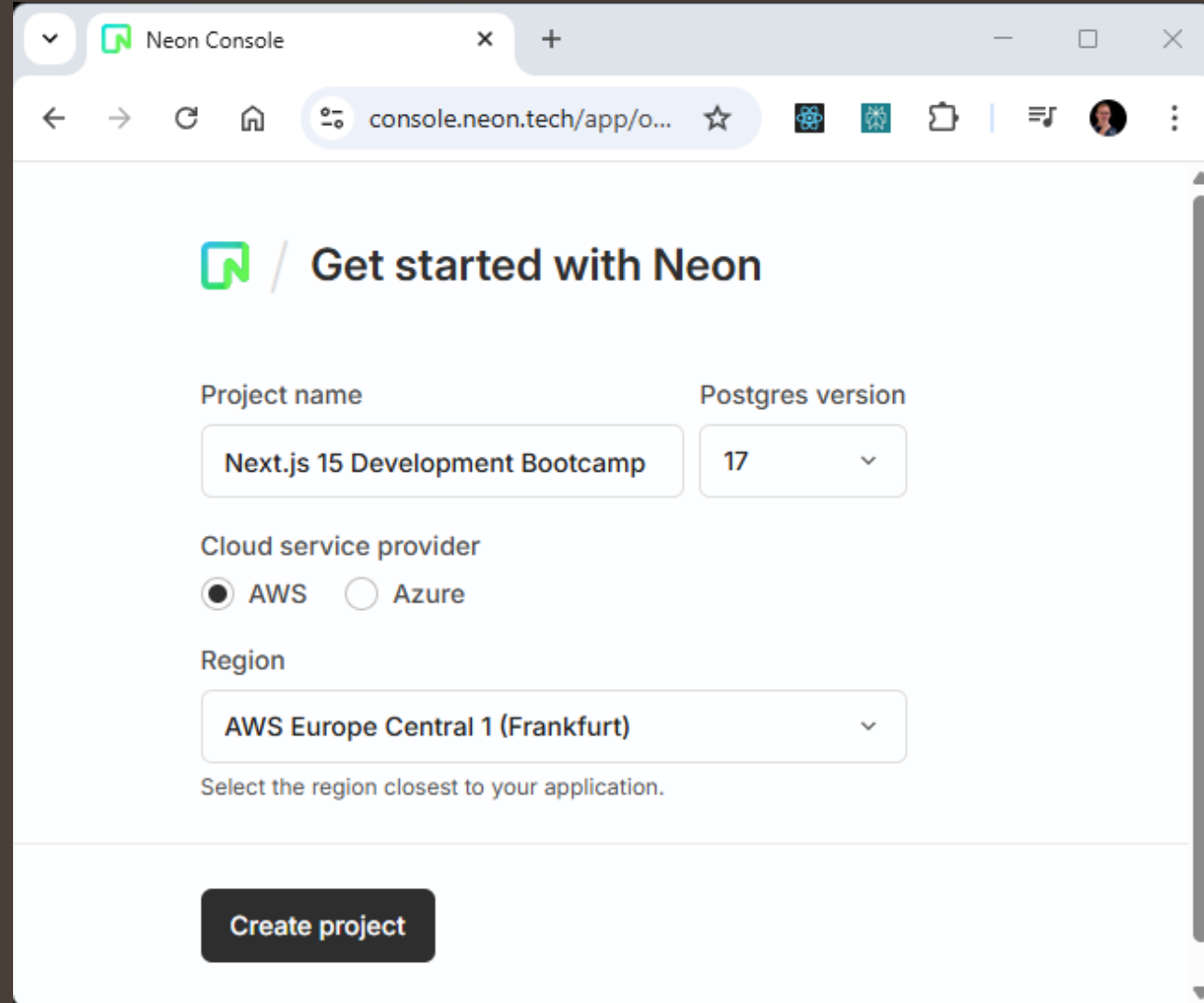
# Using Neon

- **Serverless Architecture**
  - Neon provides a serverless PostgreSQL platform that automatically scales compute resources up and down based on demand, eliminating the need to manage server infrastructure or worry about capacity planning.
- **Instant Database Creation**
  - You can create a new PostgreSQL database in seconds through Neon's web console or CLI, with no complex setup or configuration required - just sign up and start building immediately.
- **Branching for Development**
  - Neon offers Git-like database branching that allows you to create isolated database copies for development, testing, and staging environments without duplicating data storage costs.
- **Generous Free Tier**
  - Neon provides a substantial free tier with 512 MB of storage and 1 compute unit, making it excellent for development projects, prototypes, and small applications without upfront costs.
- **Standard PostgreSQL Compatibility**
  - Since Neon runs genuine PostgreSQL, you can use all your existing tools, extensions, and SQL knowledge without any vendor lock-in or proprietary syntax to learn.

# Why Neon instead of AWS

- **Serverless vs. Always-On Instances**
  - Neon automatically scales compute to zero when inactive and scales up on demand, while AWS RDS requires you to provision and pay for fixed database instances that run continuously, even during periods of no activity.
- **Simplified Management and Setup**
  - Neon handles all database administration tasks through a streamlined interface, whereas AWS RDS requires configuring VPCs, security groups, parameter groups, and other infrastructure components that add complexity to deployment and maintenance.
- **Developer-Focused Features**
  - Neon provides Git-like database branching for development workflows and instant database creation, while AWS RDS focuses on enterprise features like Multi-AZ deployments and read replicas that may be overkill for many applications.

# Deploying the database to the cloud

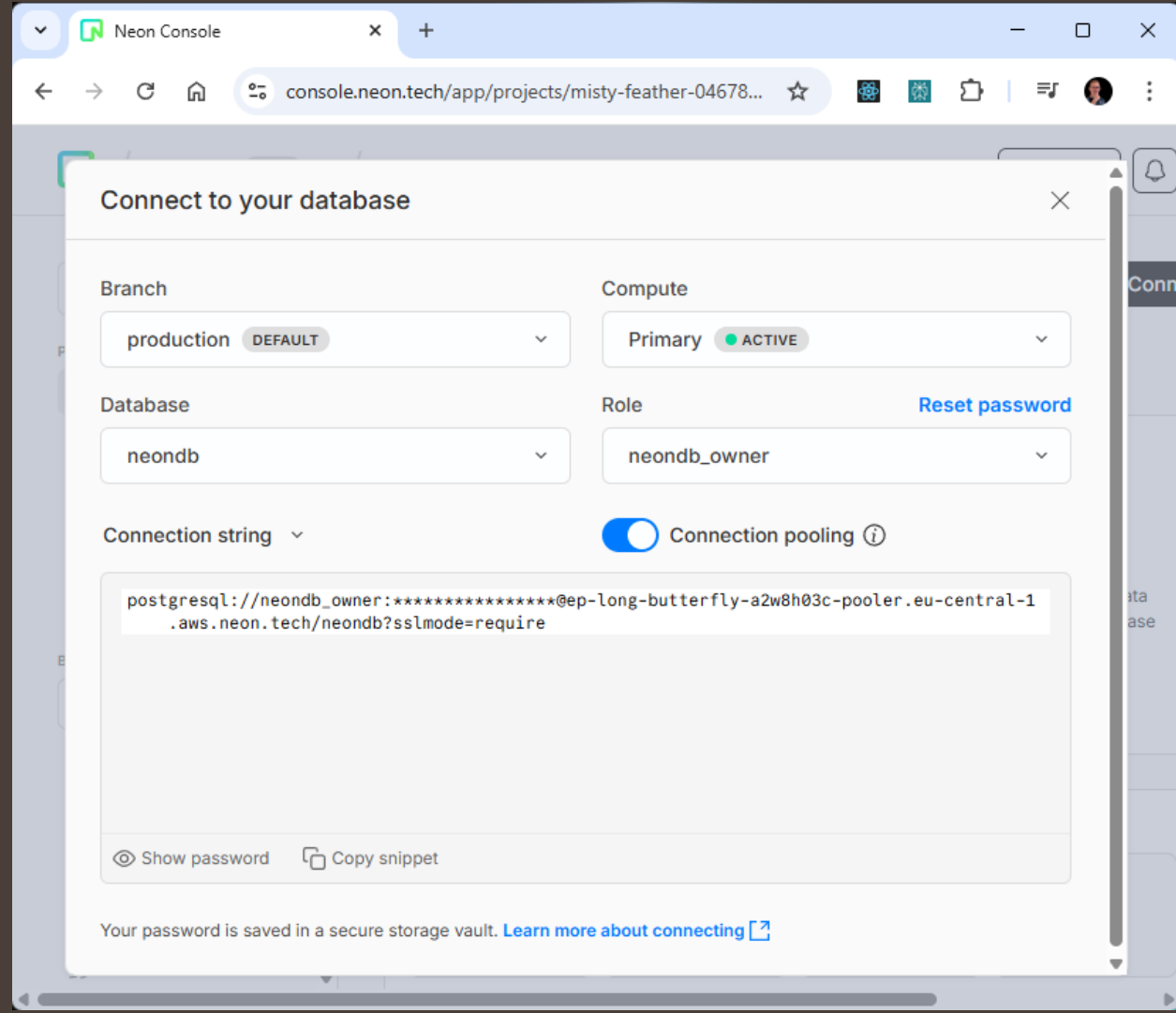


The screenshot shows the Neon Console interface in a web browser. The browser's address bar displays 'console.neon.tech/app/o...'. The page title is 'Get started with Neon'. The form contains the following fields:

- Project name:** A text input field containing 'Next.js 15 Development Bootcamp'.
- Postgres version:** A dropdown menu showing '17'.
- Cloud service provider:** Two radio buttons, with 'AWS' selected and 'Azure' unselected.
- Region:** A dropdown menu showing 'AWS Europe Central 1 (Frankfurt)'.

Below the region dropdown, there is a note: 'Select the region closest to your application.' At the bottom of the form is a dark button labeled 'Create project'.

# Deploying the database to the cloud



# Deploying the database to the cloud

```
Windows PowerShell
PS C:\demos\next-15-bootcamp> npx prisma migrate deploy
Environment variables loaded from .env
Prisma schema loaded from prisma\schema.prisma
Datasource "db": PostgreSQL database "neondb", schema "public" at
"ep-long-butterfly-a2w8h03c-pooler.eu-central-1.aws.neon.tech"

1 migration found in prisma/migrations

Applying migration `20250602192938_movies`

The following migration(s) have been applied:

migrations/
├─ 20250602192938_movies/
└─ migration.sql

All migrations have been successfully applied.
PS C:\demos\next-15-bootcamp> |
```

# Deploying the database to the cloud



```
Windows PowerShell
PS C:\demos\next-15-bootcamp> npx prisma db seed
Environment variables loaded from .env
Running seed command `tsx ./prisma/seed.ts` ...

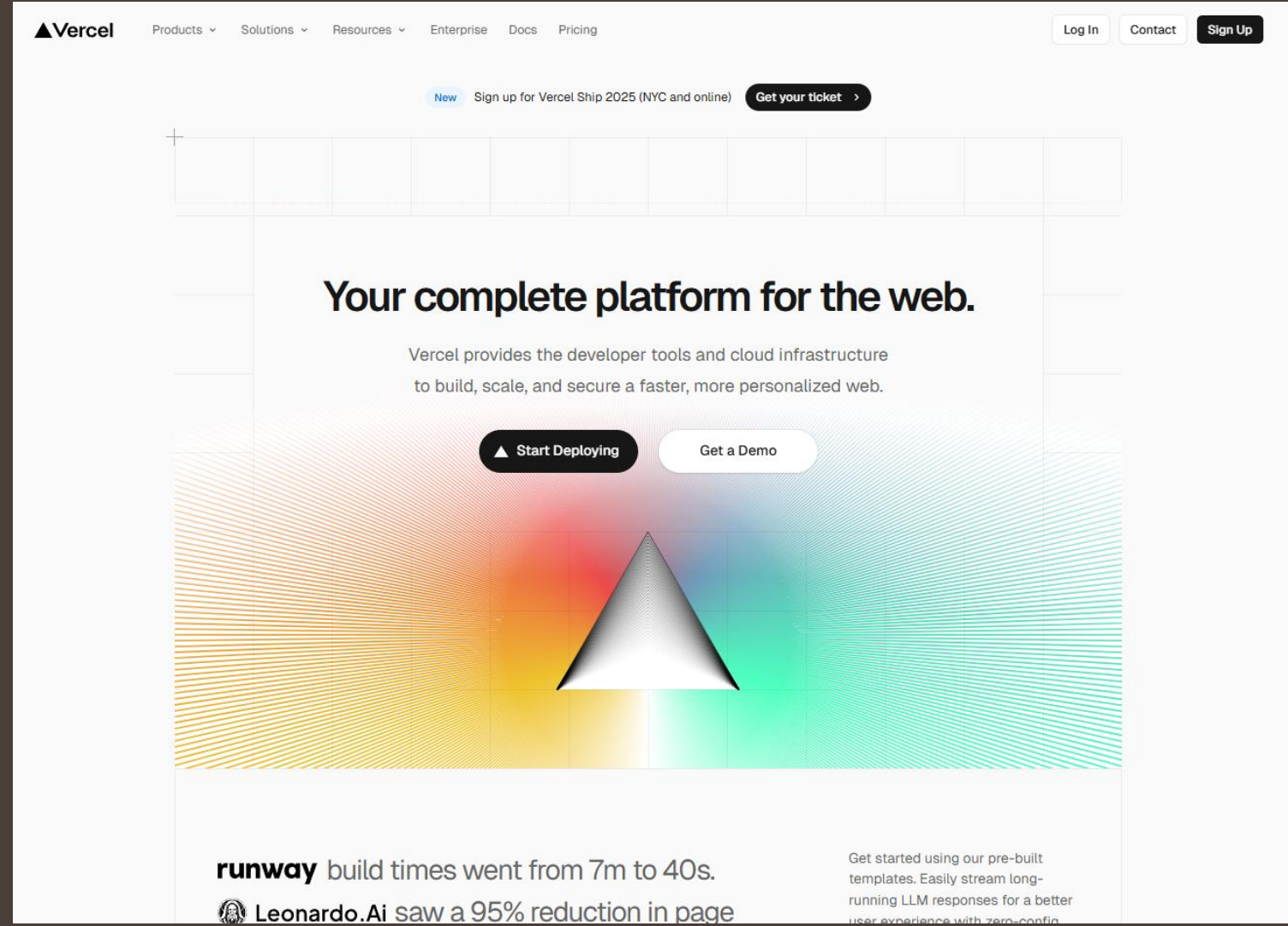
The seed command has been executed.
```

# Deploying the application to Vercel

# Deploying the application

- **Next.js applications can be self hosted and deployed anywhere**
  - Provided you have a Node.js runtime
  - Just do `npm run build && npm start`
- **Self hosting works well in a Docker container**
  - Great portability and platform agnostic
  - Runs anywhere Docker is supported
  - Avoids vendor lock-in
- **Many providers make this even easier and better using adapters**
  - AWS Amplify
  - Cloudflare
  - Deno Deploy
  - Netlify
  - Vercel

# Deploying the application



The screenshot shows the Vercel website homepage. At the top, the Vercel logo is on the left, and navigation links for Products, Solutions, Resources, Enterprise, Docs, and Pricing are in the center. On the right, there are buttons for Log In, Contact, and Sign Up. Below the navigation bar, a banner for 'New Sign up for Vercel Ship 2025 (NYC and online)' with a 'Get your ticket' button is displayed. The main content area features a large heading 'Your complete platform for the web.' followed by a subheading 'Vercel provides the developer tools and cloud infrastructure to build, scale, and secure a faster, more personalized web.' Below this, there are two buttons: 'Start Deploying' and 'Get a Demo'. A large, colorful, abstract graphic consisting of many thin lines radiating from a central point is positioned behind the buttons. At the bottom, there are two testimonials: 'runway build times went from 7m to 40s.' and 'Leonardo.Ai saw a 95% reduction in page', followed by a small text block about pre-built templates and LLM responses.

▲ Vercel

Products Solutions Resources Enterprise Docs Pricing

Log In Contact Sign Up


New Sign up for Vercel Ship 2025 (NYC and online) Get your ticket >

## Your complete platform for the web.

Vercel provides the developer tools and cloud infrastructure to build, scale, and secure a faster, more personalized web.

▲ Start Deploying Get a Demo

**runway** build times went from 7m to 40s.

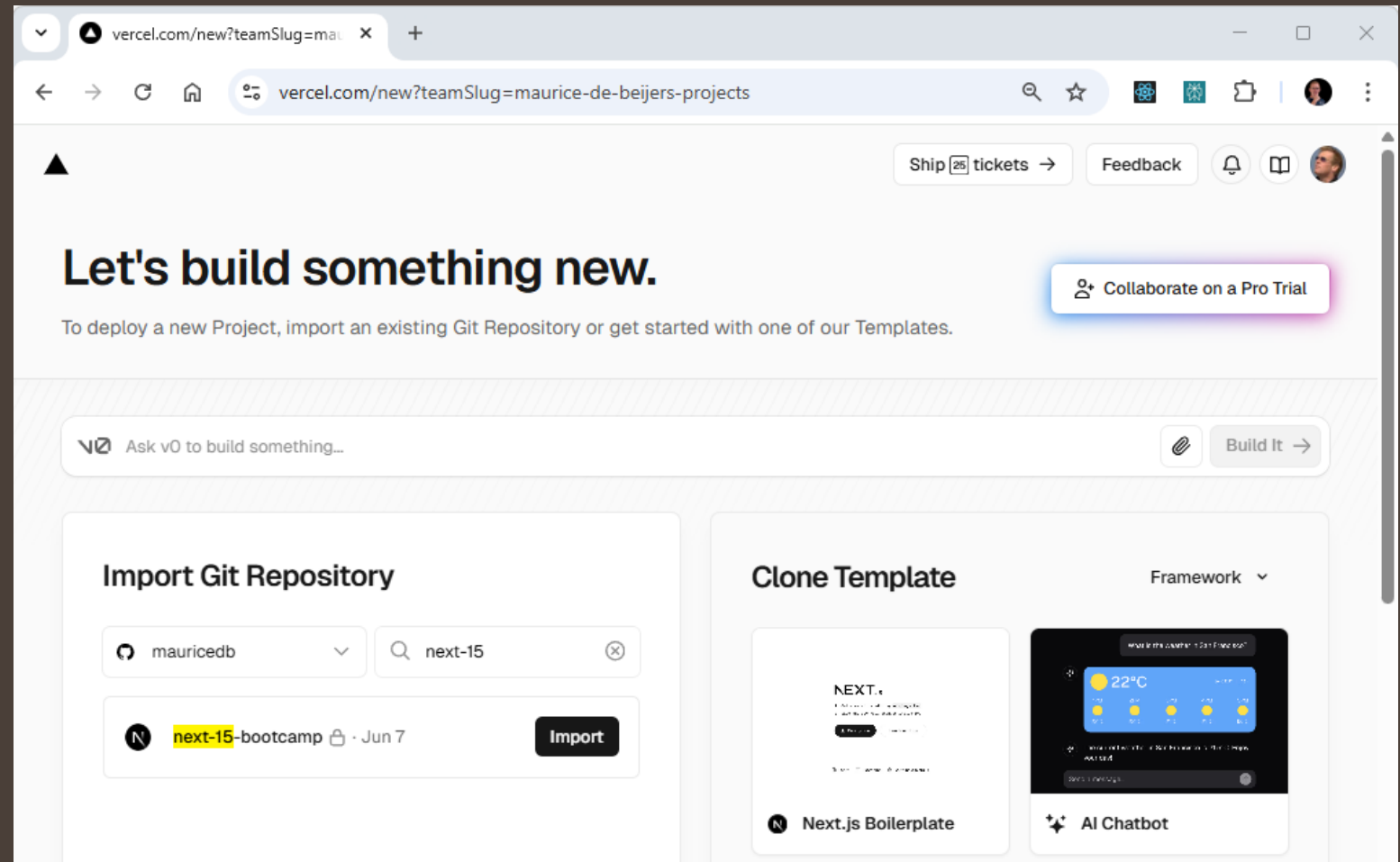
 **Leonardo.Ai** saw a 95% reduction in page

Get started using our pre-built templates. Easily stream long-running LLM responses for a better user experience with zero-config.

# Deploying the application to Vercel

- **Vercel is the company behind Next.js**
  - A significant part of the React core team work there
- They use **usage-based pricing**
  - With a with free tier
- A lot of **build in tooling and services**
  - CDN for scaling
  - Continuous deployment based on updates to GitHub's main branch
  - Automatic preview deployments with GitHub PR's

# Deploying the application to Vercel



# Deploying the application to Vercel

The screenshot shows the Vercel 'New Project' interface in a web browser. The browser's address bar shows the URL: `vercel.com/new/import?s=https%3A%2F%2Fgithub.com%2Fmauricedb%2Fnext-15...`. The page title is 'New Project - Vercel'.

The form is titled 'New Project' and indicates it is 'Importing from GitHub' for the repository `mauricedb/next-15-bootcamp` on the `main` branch.

Below this, a message says: 'Choose where you want to create the project and give it a name.'

The 'Vercel Team' dropdown is set to 'Maurice de Beijer's proj...' with a 'Hobby' plan. The 'Project Name' field contains 'next-15-bootcamp'.

The 'Framework Preset' dropdown is set to 'Next.js'.

The 'Root Directory' field contains './' with an 'Edit' button next to it.

There is a section for 'Build and Output Settings' which is currently collapsed.

Below that is an 'Environment Variables' section, also collapsed. When expanded, it shows a table with the following data:

Key	Value
DATABASE_URL	1.aws.neon.tech/neondb?sslmode=require

Below the table is a '+ Add More' button.

A tip at the bottom of the environment variables section reads: 'Tip: Paste an .env above to populate the form. [Learn more](#)'.

At the very bottom of the form is a large black button labeled 'Deploy'.

# Deploying the application to Vercel

The screenshot displays the Vercel dashboard for a project named 'next-15-bootcamp'. The interface includes a top navigation bar with the project name and a 'next-15-bootcamp' dropdown. Below this is a secondary navigation bar with links for Overview, Deployments, Analytics, Speed Insights, Logs, Observability, Firewall, Storage, Flags, AI, and Settings. The main content area features a 'next-15-bootcamp' header with buttons for Repository, Usage, Domains, and a Visit button. A 'Production Deployment' section shows a deployment named 'next-15-bootcamp-mkg15hh0k-maurice-de-beijers-projects.vercel.app' with a status of 'Ready' and a source of 'main'. Below this is a 'Deployment Configuration' section with checkboxes for Fluid Compute, Deployment Protection, and Skew Protection. At the bottom, there are three panels: 'Firewall' (active), 'Observability' (showing 52 Edge Requests and 0 Function Invocations), and 'Analytics' (showing 0 Track visitors and page views).

next-15-bootcamp - Overview

vercel.com/maurice-de-beijers-projects/next-15-bootcamp

Maurice de Beijer's projects Hobby next-15-bootcamp

Ship tickets Feedback

Overview Deployments Analytics Speed Insights Logs Observability Firewall Storage Flags AI Settings

next-15-bootcamp Repository Usage Domains Visit

Production Deployment Build Logs Runtime Logs Instant Rollback

Movie Comparer

Compare. Decide. Watch.

Find your next favorite movie with the most powerful comparison tool.

The Ultimate Movie Comparison Experience

Movie Comparer is the best app to compare different movies and make an informed choice on what to watch next. Our platform allows you to compare movies side-by-side based on ratings, reviews, cast, and more to help you decide.

Deployment

next-15-bootcamp-mkg15hh0k-maurice-de-beijers-projects.vercel.app

Domains

next-15-bootcamp.vercel.app +2

Status Created

Ready 2h ago by mauricedb

Source

main

1ff690e Fix HTML entity in movie night message for proper rendering

> Deployment Configuration

Fluid Compute Deployment Protection Skew Protection

To update your Production Deployment, push to the main branch.

Deployments

Firewall 24h Enable Bot Protection

Firewall is active

Observability 6h

Edge Requests 52

Function Invocations 0

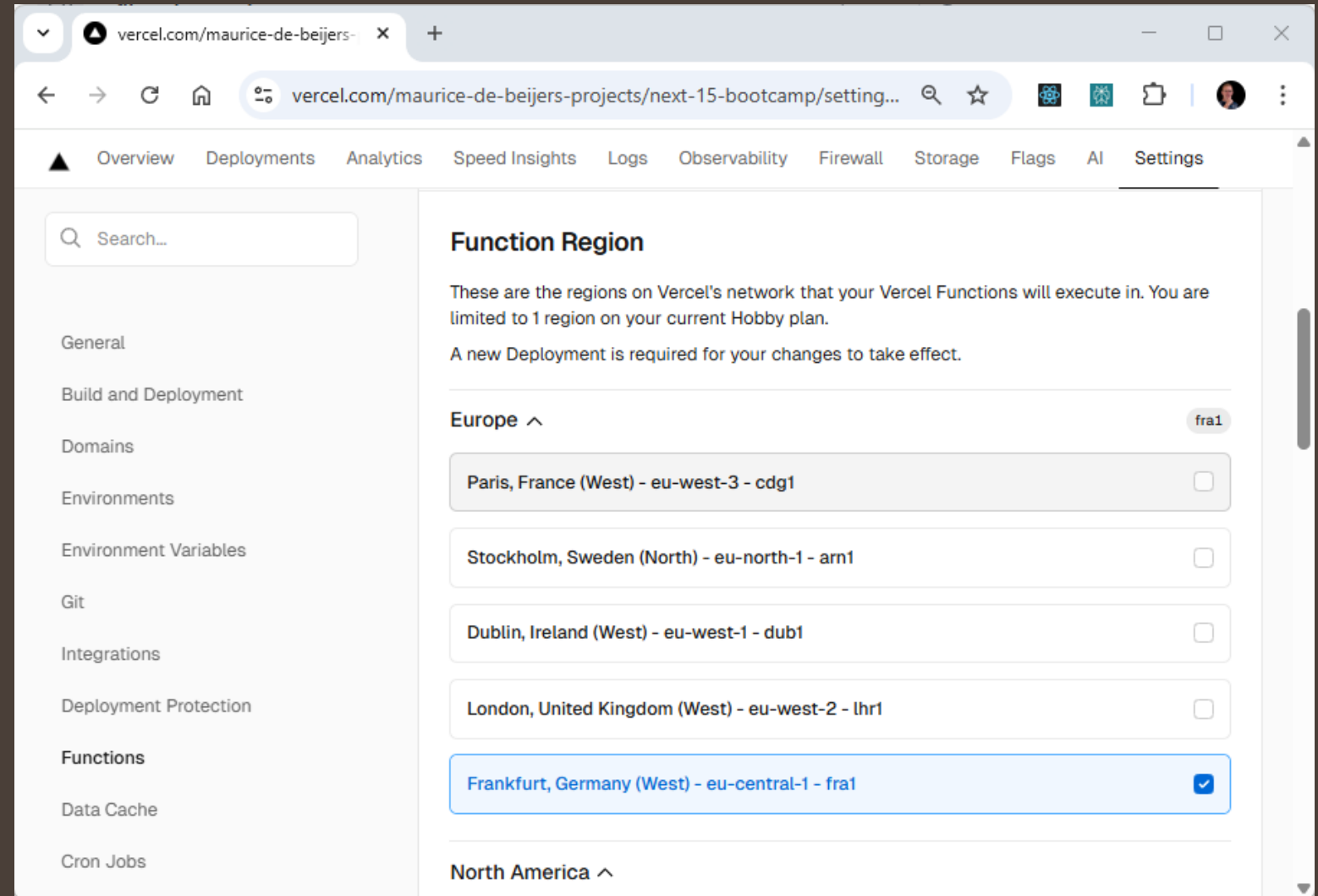
Error Rate 0%

Analytics

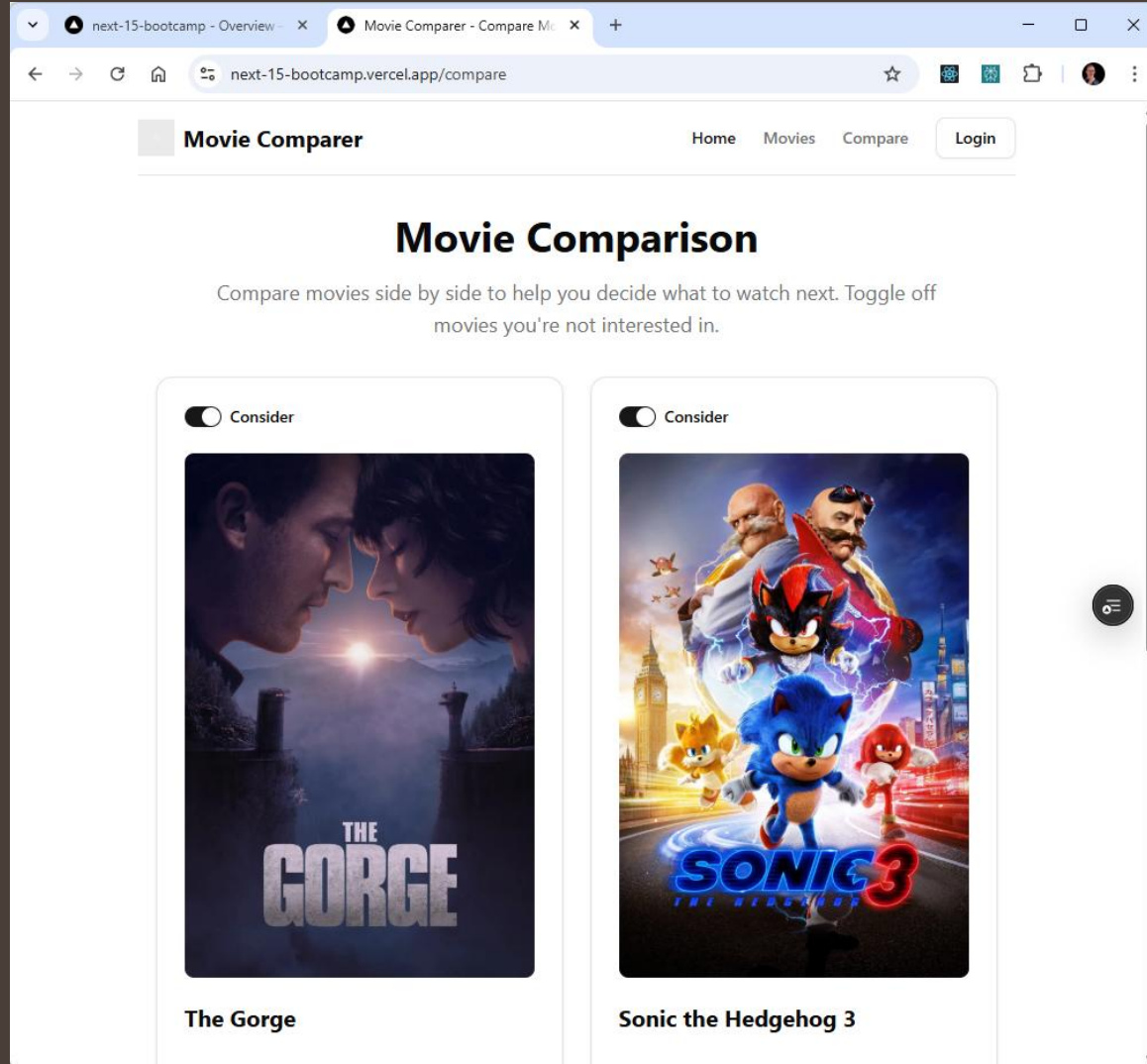
Track visitors and page views


Enable

# Deploying the application to Vercel



# Deploying the application





Persisting selected  
movies

# Persisting selected movies

- Automatic State Synchronization
  - Valtio's `subscribe()` function reacts to any state change in your proxy object, automatically triggering `localStorage` saves whenever selected movies are added or removed without requiring manual persistence calls throughout your application code.
- Simple JSON Serialization Pattern
  - You can initialize your state from `localStorage` with `JSON.parse(localStorage.getItem('selectedMovies'))` and then use `subscribe()` to automatically save the entire state object with `JSON.stringify(state)` whenever it changes, creating a seamless two-way sync.
- Minimal Setup Code
  - The entire persistence setup takes just a few lines of code - create your proxy state with initial `localStorage` data, then add one `subscribe()` call that handles all future saves automatically, eliminating complex persistence logic scattered throughout your components.
- Real-time Persistence
  - Since `subscribe()` triggers on every state mutation, your selected movies are immediately saved to `localStorage` as soon as users select or deselect them, ensuring no data loss even if the browser crashes or the user accidentally closes the tab.
- Zero Component Coupling
  - The persistence logic lives entirely outside your React components, so you can add, remove, or modify selected movies from any component without worrying about remembering to save the state - the `subscribe()` function handles it automatically in the background.

# Persisting selected movies



```
File Edit Selection View Go Run Terminal Help
TS store.ts IM, M x
src > lib > TS store.ts

20 export const movieStore = proxy<MovieStore>(
21   JSON.parse(
22     (typeof window !== 'undefined' && localStorage.getItem('movieStore')) ||
23     JSON.stringify(initialState)
24   )
25 );
26
27 subscribe(movieStore, () => {
28   if (typeof window !== 'undefined') {
29     localStorage.setItem('movieStore', JSON.stringify(movieStore));
30   }
31 });
```

# Conclusion

- **Next.js** is a great React application framework
- Creating a new **Next.js** application is easy
- Using **Vercel V0** can speed development a lot
- **Docker** is great for development purposes
  - Not just for deployment to production
- The **Prisma ORM** makes database access really easy
- **React sever component** makes using server data much easier
- **React client components** make client-side state easier
- **Vercel** is a great hosting platform
  - But not a requirement

Thank you for joining

Share your thoughts

