

Enterprise Starter

check

repo or workflow not found

license

repo not found

contributors

repo not found

404

badge not found

Welcome to the Next.js Enterprise Starter, a template for enterprise projects! It's loaded with features that'll help you build a high-performance, maintainable, and enjoyable app. We've done all the heavy lifting for you, so sit back, relax, and get ready to conquer the world with your incredible app! 🌍

Features

With this template, you get all the awesomeness you need:

- 🚀 **Next.js** - Fast by default, with config optimized for performance (with **App Directory**)
- 🗄️ **Prisma** - Next-generation ORM for Node.js and TypeScript
- 🎨 **Tailwind CSS** - A utility-first CSS framework for rapid UI development
- ✨ **Extremely Strict ESLint** and **Biome** - For clean, consistent, and error-free code
- 🔧 **Extremely strict TypeScript** - With **ts-reset** library for ultimate type safety
- 🚀 **GitHub Actions** - Pre-configured actions for smooth workflows, including Bundle Size and performance stats
- 🏆 **Perfect Lighthouse score** - Because performance matters
- 🔍 **Bundle analyzer plugin** - Keep an eye on your bundle size
- 🧪 **React Testing Library** - For rock-solid unit and integration tests
- ⚡ **Vitest** - Blazing fast unit testing framework
- 📝 **Playwright** - Write end-to-end tests like a pro
- 📖 **Storybook** - Create, test, and showcase your components
- 🚬 **Smoke Testing** and **Acceptance Tests** - For confidence in your deployments
- 🔄 **Conventional commits git hook** - Keep your commit history neat and tidy
- 🔭 **Observability** - Open Telemetry integration for seamless monitoring
- 🍝 **Absolute imports** - No more spaghetti imports
- 🏥 **Health checks** - Kubernetes-compatible for robust deployments
- 🏠 **Radix UI** - Headless UI components for endless customization
- 🏗️ **ShadCN UI** - Customizable component library built with Tailwind CSS
- 🏠 **CVA** - Create a consistent, reusable, and atomic design system
- 🔄 **Renovate BOT** - Auto-updating dependencies, so you can focus on coding
- 📊 **Components coupling and cohesion graph** - A tool for managing component relationships
- 🗨️ **Automated ChatGPT Code Reviews** - Stay on the cutting edge with AI-powered code reviews!
- 📄 **Semantic Release** - For automatic changelog generation and release management
- 🔍 **Knip** - Detect unused files, dependencies, and exports to keep your project clean and lean
- 📄 **Typedoc** - convert comments in TypeScript source code into rendered HTML documentation
- 🌱 **T3 Env** - Manage your environment variables with ease
- ☁️ **Gitpod** - Instant development environments in the cloud
- 🐳 **Docker** - Containerize your applications for consistency across environments
- 🔍 **Oxlint** - Advanced linting to catch more issues before they become problems
- 🏠 **Jotai** - Primitive and flexible state management for React
- 🔍 **React Query** - Powerful data fetching and caching for React
- 🔄 **TRPC** - End-to-end typesafe APIs made easy

- **Trigger.dev** - Automate your workflows with powerful triggers and actions
- **SendGrid** - Reliable email delivery for your application
- **AI Integration** - Enhance your application with cutting-edge AI capabilities

With this robust stack, you're equipped to build high-performance, maintainable, and scalable applications that leverage the latest in web technologies and best practices. From development to deployment, each tool and framework in this setup is chosen to ensure a smooth, efficient, and enjoyable development experience. Happy coding! 🚀

Table of Contents

- [Next.js Enterprise Boilerplate](#)
 - [Features](#)
 - [Table of Contents](#)
 - [Getting Started](#)
 - [Deployment](#)
 - [Scripts Overview](#)
 - [Coupling Graph](#)
 - [Testing](#)
 - [Running Tests](#)
 - [Acceptance Tests](#)
 - [Smoke Testing](#)
 - [Styling and Design System](#)
 - [CVA - A New Approach to Variants](#)
 - [State Management](#)
 - [Zustand](#)
 - [Jotai](#)
 - [Recoil](#)
 - [ChatGPT Code Review](#)
 - [Environment Variables handling](#)
 - [Contribution](#)
 - [Support](#)
 - [License](#)
 - [Contributors](#)

Getting Started

To get started with this starter, follow these steps:

1. Fork & clone repository:

```
## Don't forget to ★ star and fork it first :)  
git clone https://github.com/<your_username>/enterprise-starter.git
```

2. Install the dependencies:

```
brew install gcc  
brew reinstall vips  
pnpm install --frozen-lockfile
```

3. Run the development server:

```
pnpm dev
```

4. Open <http://localhost:3000> with your browser to see the result.

5. This project uses a git hook to enforce [conventional commits](#). To install the git hook, run the following command in the root directory of the project:

```
brew install pre-commit  
pre-commit install -t commit-msg
```

Deployment

Easily deploy your Next.js app with [Vercel](#) by clicking the button below:



Scripts Overview

The following scripts are available in the [package.json](#):

- **dev**: Starts the development server with all the necessary features for a great dev experience.
- **build**: Builds the application for production.
- **start**: Starts the application in production mode.
- **lint**: Lints the code using ESLint, Biome, Oxlint.
- **test**: Runs the tests using Vitest.
- **e2e**: Runs the end-to-end tests using Playwright.
- **storybook**: Starts Storybook for building UI components in isolation.
- **analyze**: Analyzes the bundle size using Webpack Bundle Analyzer.

For more detailed usage instructions and additional scripts, please refer to the [package.json](#) file.

Coupling Graph

The coupling graph is a powerful tool for visualizing and managing the relationships between components in your project. It helps you understand how different components interact with each other, identify potential areas for improvement, and ensure a maintainable and scalable architecture.

Here's how you can generate and view the coupling graph for your project:

1. Install the necessary dependencies for generating the graph:

```
pnpm install
```

2. Run the script to generate the coupling graph:

```
pnpm run generate-coupling-graph
```

3. Open the generated graph in your browser:

```
open path/to/generated/graph.html
```

The coupling graph provides a visual representation of the relationships between components in your project, making it easier to identify potential issues and areas for improvement. By analyzing the graph, you can gain insights into the coupling and cohesion of your components, and take steps to improve the overall architecture of your project.

Testing

Testing is a crucial part of the development process, and this project comes with a comprehensive testing setup to ensure the highest quality of code. The following testing tools and frameworks are included:

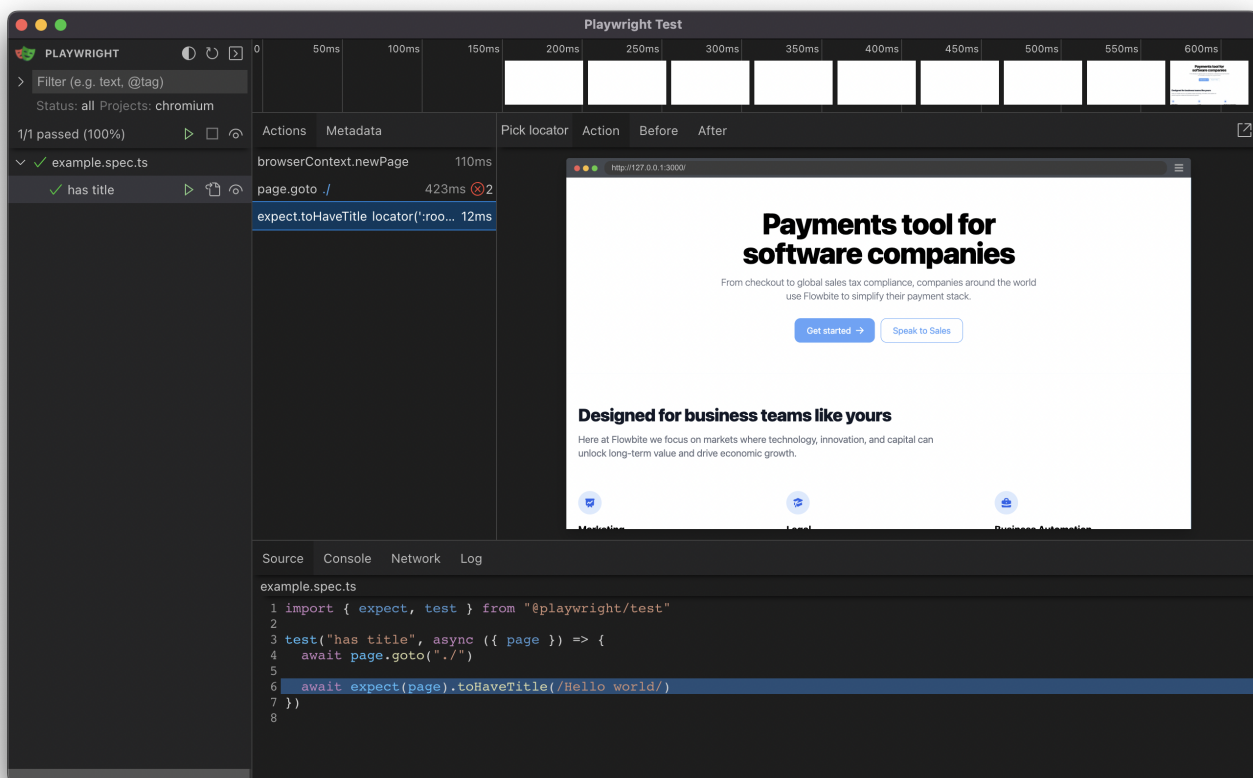
- **React Testing Library:** Helps you test React components in a way that mimics how users interact with them.
- **Vitest:** A fast unit-testing framework.
- **Playwright:** End-to-end testing framework.

Running Tests

To run the tests, simply use the following command:

```
pnpm test
```

- **Unit and integration tests:** Run Vitest tests using `pnpm test`
- **End-to-end tests (headless mode):** Run Playwright tests in headless mode with `pnpm e2e:headless`
- **End-to-end tests (UI mode):** Run Playwright tests with UI using `pnpm e2e:ui`



Acceptance Tests

To write acceptance tests, we leverage Storybook's [play function](#). This allows you to interact with your components and test various user flows within Storybook.

```

/*
 * See https://storybook.js.org/docs/react/writing-stories/play-
function#working-with-the-canvas
 * to learn more about using the canvasElement to query the DOM
 */
export const FilledForm: Story = {
  play: async ({ canvasElement }) => {
    const canvas = within(canvasElement);

    const emailInput = canvas.getByLabelText('email', {
      selector: 'input'
    });

    await userEvent.type(emailInput, 'example-email@email.com', {
      delay: 100
    });

    const passwordInput = canvas.getByLabelText('password', {
      selector: 'input'
    });

    await userEvent.type(passwordInput, 'ExamplePassword', {
      delay: 100
    });
  }
};

```

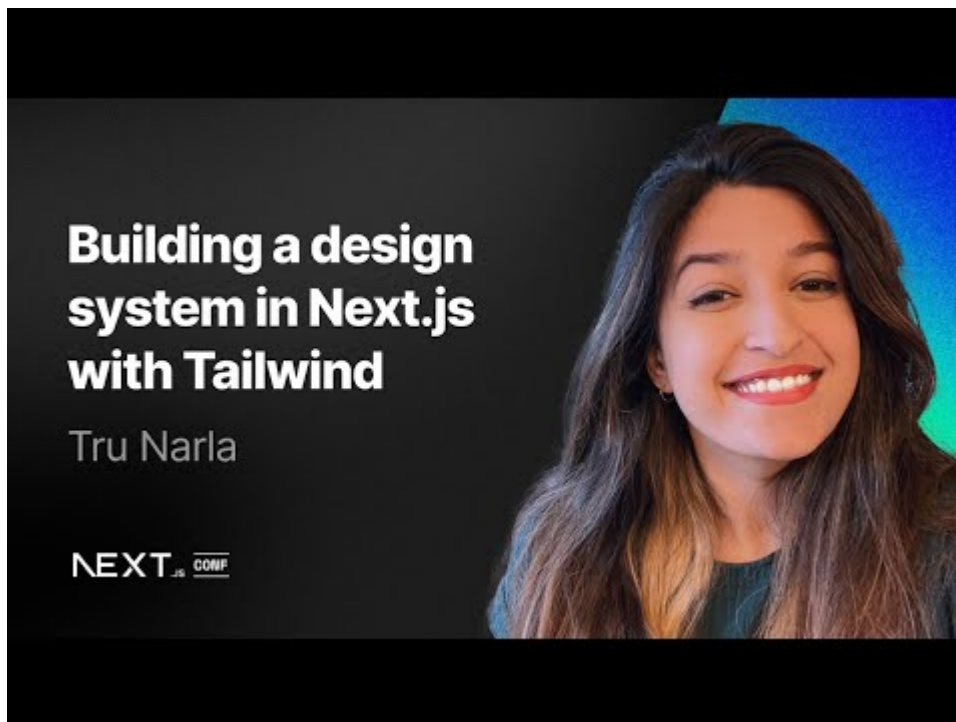
```
});  
// See  
https://storybook.js.org/docs/react/essentials/actions#automatically-  
matching-args to learn how to setup logging in the Actions panel  
const submitButton = canvas.getByRole('button');  
  
await userEvent.click(submitButton);  
}  
};
```

Smoke Testing

In this starter, we use Storybook's out-of-the-box support for smoke testing to verify that components render correctly without any errors. Just run `pnpm test-storybook` to perform smoke testing. Remember to write stories in JSX or TSX format only. Smoke testing and a lot of other functionalities don't work well with MDX stories.

🎨 Styling and Design System

This starter uses Tailwind CSS for styling and CVA for creating a powerful, easy-to-use design system. If you want to learn more about the setup, check out this fantastic video by Vercel:



CVA - A New Approach to Variants

While CSS-in-TS libraries such as [Stitches](#) and [Vanilla Extract](#) are great for building type-safe UI components, they might not be the perfect fit for everyone. You may prefer more control over your stylesheets, need to use a framework like Tailwind CSS, or simply enjoy writing your own CSS.

Creating variants using traditional CSS can be a tedious task, requiring you to manually match classes to props and add types. CVA is here to take that pain away, allowing you to focus on the enjoyable aspects of

UI development. By providing an easy and type-safe way to create variants, CVA simplifies the process and helps you create powerful design systems without compromising on the flexibility and control of CSS.

State Management

While this starter doesn't include a specific state management library, we believe it's essential for you to choose the one that best suits your project's needs. Here are some libraries we recommend for state management:

Zustand

[Zustand](#) is a small, fast, and scalable state management library. It's designed to be simple and intuitive, making it a great choice for small to medium-sized projects. It's also optimized for bundle size, ensuring minimal impact on your app's performance.

Jotai

[Jotai](#) is an atom-based state management library for React that focuses on providing a minimal and straightforward API. Its atom-based approach allows you to manage your state in a granular way while still being highly optimized for bundle size.

Recoil

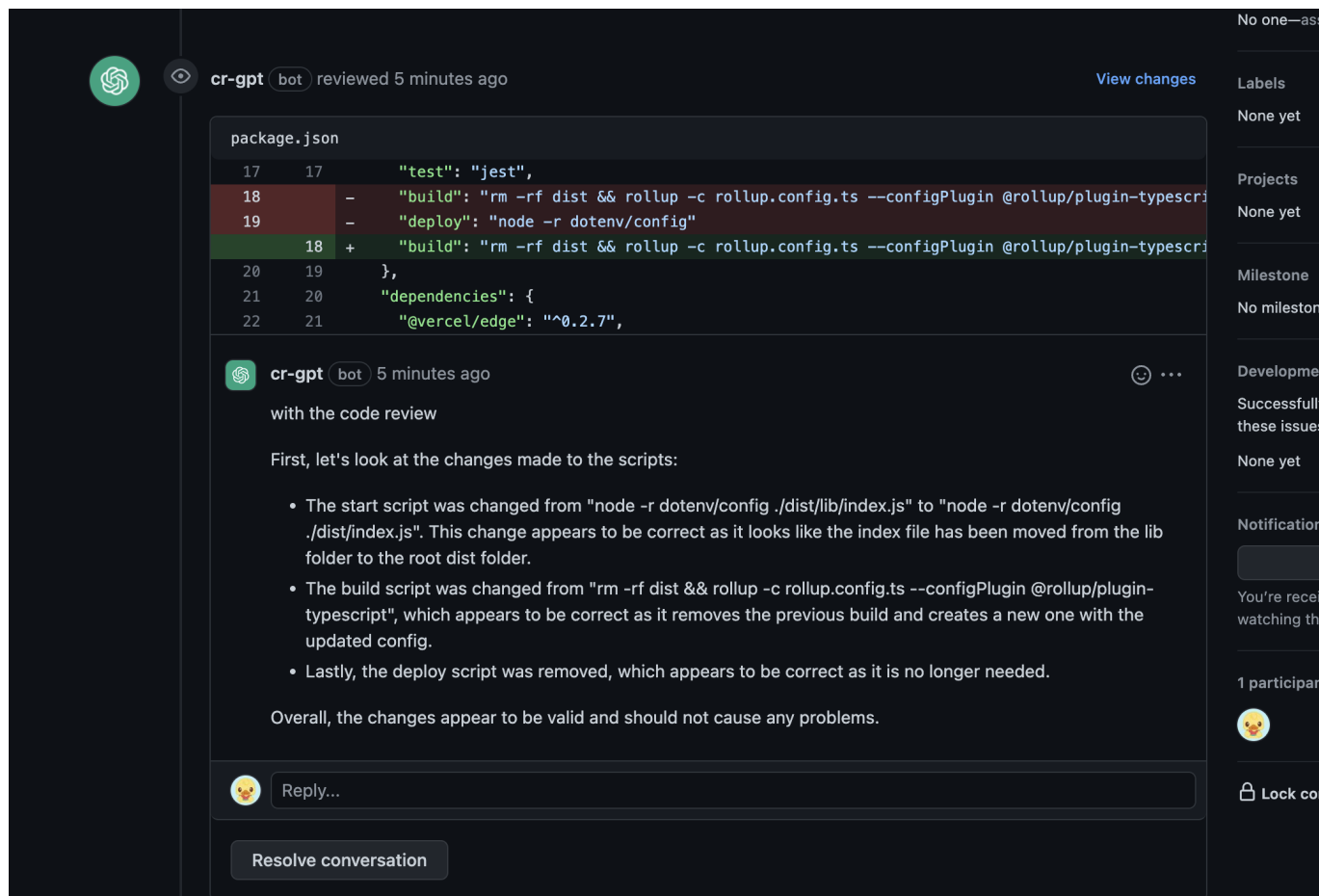
[Recoil](#) is a state management library developed by Facebook, specifically designed for React applications. By utilizing atoms and selectors, Recoil allows you to efficiently manage state and derived state. Its key benefit is the ability to update components only when the state they're subscribed to changes, reducing unnecessary re-renders and keeping your application fast and efficient. Recoil also offers great developer experience with built-in debugging tools.

Choose the library that best fits your requirements and project structure to ensure an efficient state management solution for your application.

ChatGPT Code Review

We've integrated the innovative [ChatGPT Code Review](#) for AI-powered, automated code reviews. This feature provides real-time feedback on your code, helping improve code quality and catch potential issues.

To use ChatGPT Code Review, add an `OPENAI_API_KEY` environment variable with an appropriate key from the OpenAI platform. For setup details, refer to the [Using GitHub Actions](#) section in the documentation.



The screenshot shows a GitHub pull request interface. At the top, a bot named 'cr-gpt' reviewed the changes 5 minutes ago. The review is for a pull request titled 'package.json'. The diff shows changes to the 'package.json' file, specifically the 'build' and 'dependencies' sections. The 'build' script was updated to 'rm -rf dist && rollup -c rollup.config.ts --configPlugin @rollup/plugin-typescript' and the 'dependencies' section was updated to include '@vercel/edge' with a version constraint of '^0.2.7'.

The review text from the bot is as follows:

with the code review

First, let's look at the changes made to the scripts:

- The start script was changed from "node -r dotenv/config ./dist/lib/index.js" to "node -r dotenv/config ./dist/index.js". This change appears to be correct as it looks like the index file has been moved from the lib folder to the root dist folder.
- The build script was changed from "rm -rf dist && rollup -c rollup.config.ts --configPlugin @rollup/plugin-typescript", which appears to be correct as it removes the previous build and creates a new one with the updated config.
- Lastly, the deploy script was removed, which appears to be correct as it is no longer needed.

Overall, the changes appear to be valid and should not cause any problems.

At the bottom of the review, there is a 'Reply...' input field and a 'Resolve conversation' button.

Environment Variables handling

T3 Env is a library that provides environmental variables checking at build time, type validation and transforming. It ensures that your application is using the correct environment variables and their values are of the expected type. You'll never again struggle with runtime errors caused by incorrect environment variable usage.

Config file is located at **env.mjs**. Simply set your client and server variables and import **env** from any file in your project.

```
export const env = createEnv({
  server: {
    // Server variables
    SECRET_KEY: z.string()
  },
  client: {
    // Client variables
    API_URL: z.string().url()
  },
  runtimeEnv: {
    // Assign runtime variables
    SECRET_KEY: process.env.SECRET_KEY,
    API_URL: process.env.NEXT_PUBLIC_API_URL
  }
});
```


If the required environment variables are not set, you'll get an error message:

✖ Invalid environment variables: { SECRET_KEY: ['Required'] }

🤝 Contribution

Contributions are always welcome! To contribute, please follow these steps:

1. Fork the repository.
2. Create a new branch with a descriptive name.
3. Make your changes, and commit them using the [Conventional Commits](#) format.
4. Push your changes to the forked repository.
5. Create a pull request, and we'll review your changes.