



Welcome to Teor documentation

Table of contents

Quick Start Guide	2
Getting Started	2
Installation and System Requirements	2
Live Demo Access Credentials	6
Key Features	6
Troubleshooting	12
API Documentation	13
Best Practices	13
Support	15

Thank you for choosing the Teor! This documentation provides all the essential information and guidance to use the template.

Whether you are a new user or an experienced seeking detailed insights, this documentation is designed to cater to your needs.



Quick Start Guide

1. Ensure you have the Node.js development environment ready. Also, check Node.js 16.8, MySQL, and CLI access available.
2. Extract the Teor zip file and then rename the file “.env.example” to “.env” then update all the necessary variables.
3. Update the *next.config.js* file with your Cloudinary setup info.
4. Navigate to the project’s root directory in your preferred command prompt or terminal.
5. Run the command: **npm install**
6. Once the installation is done, run **npm run dev**
7. Browse the URL on the browser: <http://localhost:3000>

Getting Started

To begin your journey with Teor, we recommend starting with the “Getting Started” section. Here, you will find step-by-step instructions on how to set up, configure, and use the core features. This section is ideal for buyers who have bought the Teor.

Installation and System Requirements

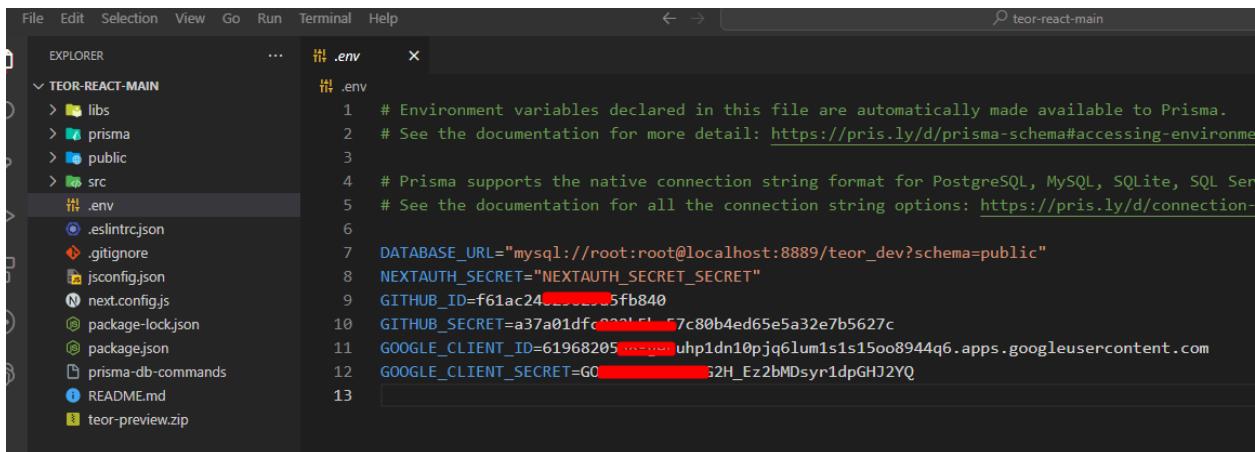
Before diving into the features, ensure your system meets Teor requirements. The “Installation and System Requirements” section comprehensively overviews the hardware and software prerequisites.

System Requirements install on your machine:

- [Node.js 16.8](#) or later.
- [MySQL](#)
- MacOS, Windows (including WSL), and Linux are supported.
-

Step 1. Extract the downloaded from [ThemeForest](#) to your preferred directory.

Step 2. Rename the file named “.env.example” to either “.env” or “.env.local”, and then update all the necessary variables as follows.



```

File Edit Selection View Go Run Terminal Help
EXPLORER .env
TEOR-REACT-MAIN
libs prisma public src .env .eslintrc.json .gitignore jsconfig.json next.config.js package-lock.json package.json prisma-db-commands README.md teor-preview.zip
.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
3
4 # Prisma supports the native connection string format for PostgreSQL, MySQL, SQLite, SQL Server, and MongoDB. See the documentation for all the connection string options: https://pris.ly/d/connection-strings
5
6
7 DATABASE_URL="mysql://root:root@localhost:8889/teor_dev?schema=public"
8 NEXTAUTH_SECRET="NEXTAUTH_SECRET_SECRET"
9 GITHUB_ID=f61ac24...5fb840
10 GITHUB_SECRET=a37a01dfc...57c80b4ed65e5a32e7b5627c
11 GOOGLE_CLIENT_ID=61968205...uhp1dn10pjg6lum1s1s15oo8944q6.apps.googleusercontent.com
12 GOOGLE_CLIENT_SECRET=G0...52H_Ez2bMDsyr1dpGHJ2YQ
13

```

- **DATABASE_URL=MySQL Database String**

- For Example During my Development:

```
DATABASE_URL="mysql://root:root@localhost:8889/teor_dev?schema=public"
```

- **NEXTAUTH_SECRET=Your NextAuth Secret**

- For Example During my Development:

```
NEXTAUTH_SECRET="arandomstringvalue"
```

- **GITHUB_ID=Your GitHub ID**

- For Example During my Development:

```
GITHUB_ID=GITHUBIDISLIKEccb65r3
```

- **GITHUB_SECRET=Your GitHub Secret**

- For Example During my Development:

```
GITHUB_SECRET=GITHUBSECRETISLIKEccb65343r3
```



- GOOGLE_CLIENT_ID="[YOUR GOOGLE CLIENT ID](#)"

- For Example During my Development

`GOOGLE_CLIENT_ID=GOOGLE_CLIENT_ID_IS_LIKE_ghj789.apps.googleusercontent.com`

- GOOGLE_CLIENT_SECRET="[YOUR GOOGLE CLIENT SECRET](#)"

- For Example During my Development

GOOGLE CLIENT SECRET=GOCSPX-89jn

Step 3. Update with your Cloudinary cloud name and Preset



next.config.js

NEXT_PUBLIC_CLOUDINARY_CLOUD_NAME="[YOUR_CLOUDINARY_CLOUD_NAME](#)"

- For Example

NEXT_PUBLIC_CLOUDINARY_CLOUD_NAME="cloud-name"

NEXT_CLOUDINARY_PREST="[prsetname](#)"

- For Example

NEXT_CLOUDINARY_PRESET="preset-name"

Step 4. Create a Database on the local server ([WAMP](#) or [XAMPP](#)).

- Look for the .sql file among the files and folders. The .sql file usually has a name related to the project or database, such as “teor_dev.sql”.
- **Note:** The .sql file typically contains the SQL code needed to create the database schema and may include sample data or tables necessary for this project.
- Open the phpMyAdmin and create a new database name like “teor_dev”.
- You must update the DATABASE_URL value to reflect the newly created database. Locate the configuration file where the DATABASE_URL variable is set, usually “.env”.

```
File Edit Selection View Go Run Terminal Help
EXPLORER .env
TEOR-REACT-MAIN .env
libs prisma public src .env
.eslintrc.json .gitignore jsonconfig.json next.config.js package-lock.json package.json prisma-db-commands README.md teor-preview.zip
.env
# Environment variables declared in this file are automatically made available to Prisma.
# See the documentation for more detail: https://pris.ly/d/prisma-schema#accessing-environment-variables-from-the-schema
#
# Prisma supports the native connection string format for PostgreSQL, MySQL, SQLite, SQL Server, MongoDB and CockroachDB.
# See the documentation for all the connection string options: https://pris.ly/d/connection-strings
#
DATABASE_URL="mysql://root:root@localhost:8889/teor_dev?schema=public"
NEXTAUTH_SECRET="NEXTAUTH_SECRET"
GITHUB_ID=f61ac24825629a5fb840
GITHUB_SECRET=a37a01dfc822b5be57c80b4ed65e5a32e7b5627c
GOOGLE_CLIENT_ID=6196820538-genuhp1dn10pjg6lum1s1s15oo8944q6.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET=60CSPX-nYz4JxG2H_Ez2bMDsyr1dpGHJ2YQ
13
```

- Now, you can import the “teor_dev.sql” file into the database and import this. This file is available in the download bundle.



Step 5. Teor Project Installation.

- Navigate to the project's root directory in your preferred command prompt or terminal.
- Ensure that you have Node.js and npm installed on your system.
- Run the command: **npm install**
 - This command will initiate the installation process and download all the required dependencies specified in the "package.json" file.
 - Wait for the installation process to complete. Npm will fetch and install the necessary packages from the npm registry.
- Once the installation is finished, you will be ready to run this project by following the command: **npm run dev**
- Browse the URL on the browser <http://localhost:3000>
- Done

Live Demo Access Credentials

<https://teor-react.envytheme.com/>

Email: admin@teor.com

Password: EnvyTheme

You can login with the credentials and check how it works Dashboard, My Listings, Add Listings, Favourites & Update Profile Info, etc.

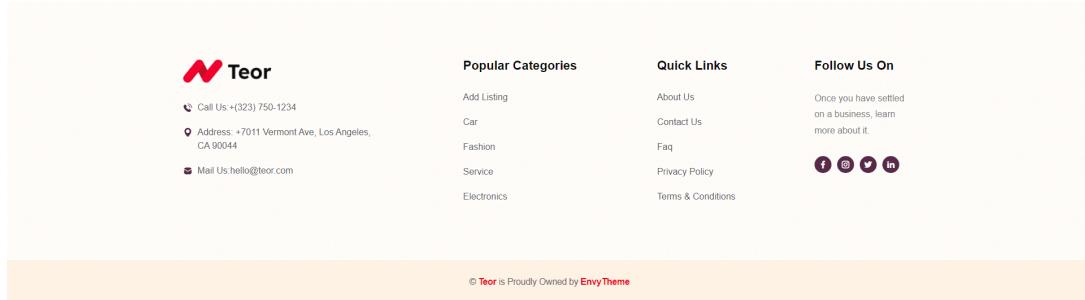
Key Features

Discover the full potential of the Teor by exploring the "Key Features" section. This part of the documentation highlights the various functionalities and tools available, empowering you to make the most out of offerings.



The main highlights of this application include:

- 1. Next.js 13 App Directory:** Utilizing the latest version of Next.js, the front-end offers exceptional performance and user experience.
- 2. Next-Auth Authentication:** Implementing Next-Auth, the application ensures robust and secure user authentication for enhanced user privacy and data protection.
<https://next-auth.js.org/>
- 3. Listings CRUD:** The application allows users to create, read, update, and delete listings, providing a seamless experience for managing their content.

A screenshot of the Teor listing creation form. The form is contained within a white box with a thin gray border. It includes fields for "Title", "Description", "Click to upload", "Anywhere", "Select Category", "Address", "Address", "Price", and a large red "Add Listing" button at the bottom. Each input field has a corresponding toolbar above it with various text and media editing icons.



Home Listing About Us Faq Blog Contact Us Teor Admin Add Listing

All listings

Dashboard Users Listings Reviews Create Blog Post

Exploring the Enchanting Wonders of Iceland

A Thriving Haven for Homebuyers and Investors

From Horse Carriages to Electric Mobility

Home Listing About Us Faq Blog Contact Us Teor Admin Add Listing

Popular Categories

Quick Links

Follow Us On

Home My Listings My Listings

My Latest Ads

Exploring the Enchanting Wonders of Iceland

A Thriving Haven for Homebuyers and Investors

From Horse Carriages to Electric Mobility

Home Listing About Us Faq Blog Contact Us Teor Admin Add Listing

Popular Categories

Quick Links

Follow Us On



4. Listings Review and Rating System:

Users can leave reviews and ratings for listings, facilitating better decision-making for other users.

The screenshot shows the Teor platform's interface. At the top, there is a navigation bar with links for Home, Listing, About Us, FAQ, Blog, and Contact Us. On the right side of the header, there is a "Teor Admin" button and a red "Add Listing" button. Below the header is a large banner image of a city street at night with many buildings and signs. Overlaid on this banner is a white "Reviews" button. The main content area has a sidebar on the left with links for Dashboard, Users, Listings, Reviews (which is highlighted in red), and Create Blog Post. The main content area displays two review cards. The first card, titled "Super :)", has a list of details: Author: Teor Admin, Date: 28 Aug 2023, Listing: A Thriving Haven for Homebuyers and Investors, and a "Delete" button. The second card, titled "Nice place indeed!", also lists details: Author: Teor Admin, Date: 6 Aug 2023, Listing: Exploring the Enchanting Wonders of Iceland, and a "Delete" button. Below this content is another screenshot of the Teor platform. It features a footer with social media links (Facebook, Instagram, Twitter, LinkedIn) and a copyright notice: "© Teor is Proudly Owned by EnvifyTheme". Above the footer is a map with a yellow pin. The main content area shows a heading "Reviews (1)" with a blue circular profile picture of a user named "Teor Admin" next to the date "6 Aug 2023". To the right of the reviews is a five-star rating icon. Below this is a form titled "Leave A Review" with a text input field labeled "Write a review" and a red "Submit Review" button.



5. Blog CRUD:

With built-in Blog CRUD functionalities, users can easily create, manage, and publish blog posts.

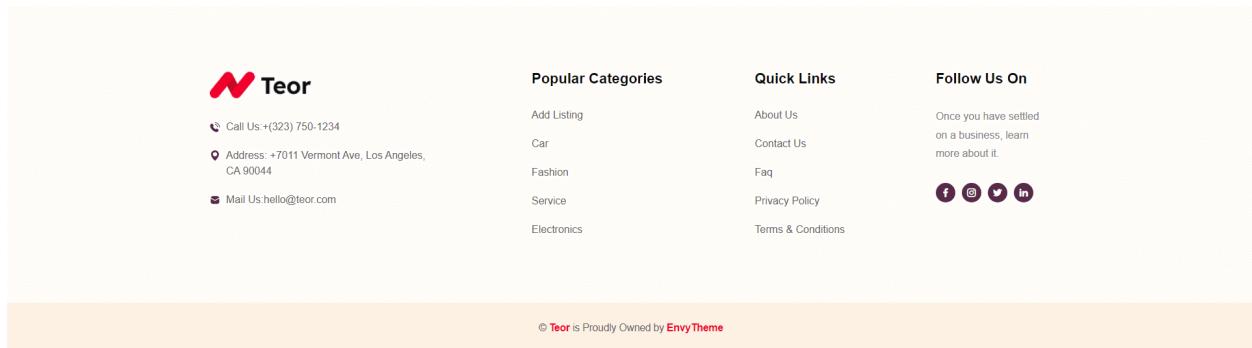


Post Title

Details Content

Select Category

Post



6. **Cloudinary Image Upload:** The application integrates with Cloudinary, enabling efficient and optimized image uploading and storage.

7. **Versatile Image Upload Options:** Users can upload images from various sources, including local computers, URLs, Google Drive, Unsplash, and Dropbox.



8. Admin Functionalities: Administrators can access special privileges and functionalities to manage users, content, and system settings.

The screenshot shows the Teor Admin Dashboard. At the top, there's a navigation bar with links for Home, Listing, About Us, Faq, Blog, Contact Us, and a 'Teor Admin' button with a user icon. Below the navigation is a large banner image of a busy city street at night. The main content area has four cards: 'Listings' (Number of listings: 21), 'Blog Posts' (Number of blog posts: 10), 'Reviews' (Number of reviews: 8), and 'Users' (Number of users: 2). On the left, a sidebar menu includes Dashboard, Users, Listings, Reviews, and Create Blog Post.

The screenshot shows the Teor Admin Users page. At the top, there's a navigation bar with links for Home, Listing, About Us, Faq, Blog, Contact Us, and a 'Teor Admin' button with a user icon. Below the navigation is a large banner image of a busy city street at night. The main content area shows a list of users: John Doe (Email: john@gmail.com) and Teor Admin (Email: admin@teor.com). Each user entry includes a small profile picture, the user's name, their email, and 'View' and 'Delete' buttons. On the left, a sidebar menu includes Dashboard, Users, Listings, Reviews, and Create Blog Post.

The screenshot shows the Teor Admin Users page. At the top, there's a navigation bar with links for Home, Listing, About Us, Faq, Blog, Contact Us, and a 'Teor Admin' button with a user icon. Below the navigation is a large banner image of a busy city street at night. The main content area shows a detailed user profile for John Doe (Email: john@gmail.com). It includes a profile picture, the user's name, their email, and 'View' and 'Delete' buttons. On the left, a sidebar menu includes Dashboard, Users, Listings, Reviews, and Create Blog Post.

The screenshot shows the Teor Admin footer. It includes a copyright notice: '© Teor is Proudly Owned by EnvyTheme'. Below it are links to 'Follow Us On' with icons for Facebook, Twitter, Instagram, and LinkedIn. The footer also contains the Teor logo and navigation links for Home, Listing, About Us, Faq, Blog, Contact Us, and a 'Teor Admin' button with a user icon.



9. And much more: The application offers other features and enhancements that contribute to its overall utility and user satisfaction.

Troubleshooting

In the unfortunate event of encountering issues while using the Teor, the “Troubleshooting and FAQs” section is here to assist you. This segment addresses common problems and provides step-by-step solutions to help you resolve issues promptly.

If you encounter an error message in the terminal or browser console, follow these steps:

- 1. Read the Error Message:** Carefully read the error message to understand the specific issue reported by the application.
- 2. Identify the Problem:** Identify the error's root cause by examining the details provided in the error message.
- 3. Google Search:** Copy the error message and search for it on Google or your preferred search engine.
- 4. Review Relevant Resources:** Look for relevant forums, documentation, or articles related to the error message to find potential solutions or explanations.
- 5. Community Support:** Consider posting the error message on developer forums or community platforms like Stack Overflow to seek help from experienced developers.
- 6. Check Dependencies:** Ensure that all dependencies are correctly installed and up-to-date, as some errors might be related to version conflicts or missing packages.
- 7. Debugging Tools:** Use debugging tools like browser developer tools to trace the error and get additional insights into the problem.



8. **Check Code Logic:** Review the relevant code section that triggered the error to identify any potential logic issues or mistakes.

9. **Trial and Error:** Experiment with different solutions to see if the error can be resolved through trial and error.

10. **Learn from Similar Issues:** If you find similar issues others face, read about their solutions to see if they apply to your situation.

Remember that error messages are valuable clues that can help you pinpoint and fix problems in the Next.js project.

API Documentation

The API documentation provides in-depth information on interacting with the services programmatically.

API routes are available under the directory "/app/api/."

Best Practices

To ensure you get the most out of the Teor, we have compiled a set of "Best Practices." These guidelines will help you optimize your workflow, avoid common pitfalls, and achieve better results.

Following best practices can lead to a more maintainable, performant, and scalable codebase when working on an ongoing Teor project. Here are some essential best practices:

1. **Folder Structure:** Organize your project into a clear and intuitive folder structure. Teor provides an excellent design structure, but you can further modularize and group related files to enhance readability and maintainability.



2. **Component Reusability:** Aim to create reusable components wherever possible. It reduces duplicate code and simplifies maintenance. Leveraging the Teor components can make your codebase more flexible and easier to manage.
3. **Data Fetching:** Handle data fetching on the server side rather than relying solely on client-side data fetching. It ensures a more reliable and consistent experience for users with slower or limited connections. The Teor project mostly used server-side for data fetching.
4. **Performance Optimization:** Pay attention to performance optimization techniques, such as lazy loading, image optimization, and minification. These can significantly improve the loading speed and overall user experience.
5. **Error Handling:** The Teor has implemented proper error handlers. Implement proper error handling mechanisms to handle exceptions and display meaningful error messages to users.
6. **Linting and Formatting:** Enforce code quality and consistency through linting and formatting tools. Use ESLint and Prettier to catch errors, enforce coding standards, and keep the codebase clean.
7. **Version Control:** Use a version control system like Git to track changes to your project over time. It allows for collaboration, easy rollback, and a solid history of your codebase.
8. **Documentation:** Maintain comprehensive and up-to-date documentation of [Next.js](#). It is vital for onboarding new team members and for future maintenance.
13. **Environment Variables:** Use environment variables to manage sensitive configuration data, such as API keys, database credentials, and other settings specific to different environments (development, staging, production).

By following the best practices, you can enhance the overall quality of the Teor project.



Support

If you are unable to find a solution to the issue, please consider creating a support ticket in our dedicated [support system](#). Our support team will promptly assist you in resolving the problem and ensuring a smooth experience with the Teor project.

We take pride in our dedicated support team. In the [Support section](#), you will find information on contacting our support team and sharing your experience with the developer.

We hope this documentation proves to be a valuable resource for your journey with the Teor. If you have any questions, feedback, or suggestions, please don't hesitate to contact us.

Happy Coding!

