```
 .d8888b.           888          888
d88P  Y88b          888          888
Y88b.               888          888
 "Y888b.   888  888 888  8888b.  888  8888b.
    "Y88b. 888  888 888     "88b 888     "88b
      "888 888  888 888 .d888888 888 .d888888
Y88b  d88P Y88b 888 888 888  888 888 888  888
 "Y8888P"   "Y88888 888 "Y888888 888 "Y888888
```

Sulala

Sulala Dashboard

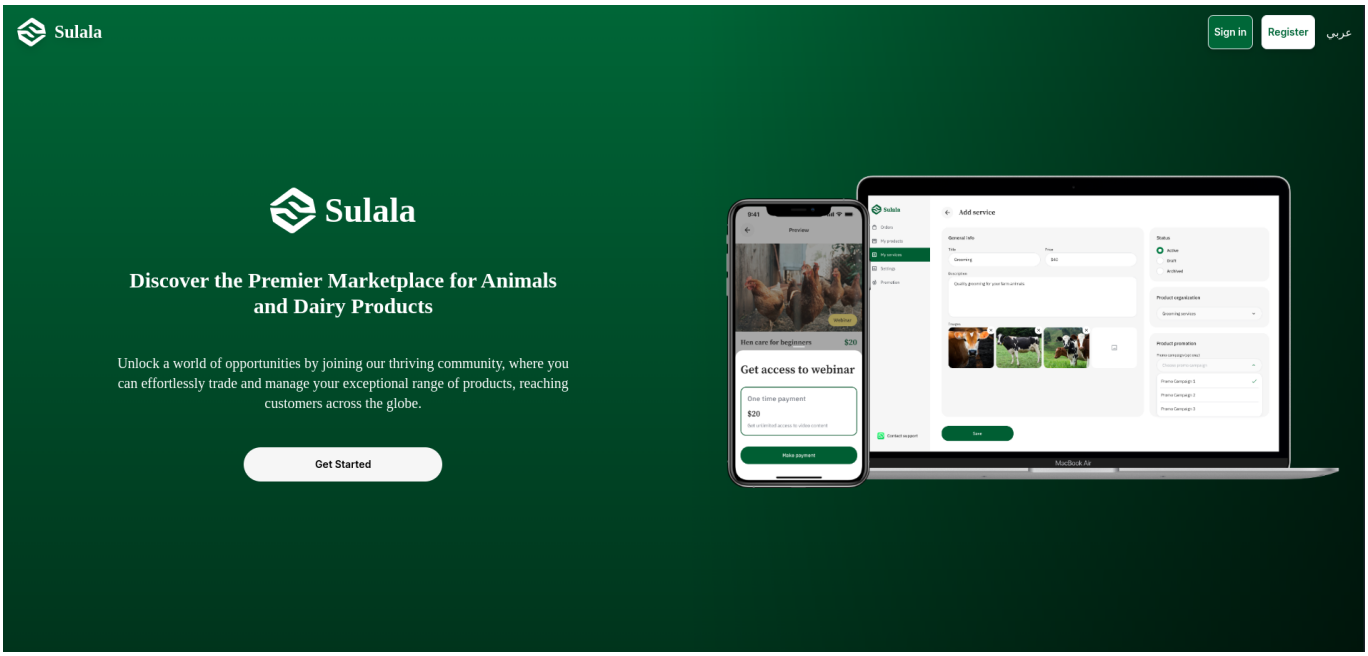
A documentation to jumpstart on this projects!

Explore the codes »

View Demo · Report Bug · Request Feature

▶ Table of Contents

About The Project



Overview

Discover Sulala, the premier Middle Eastern marketplace for animals, equipment, tools, foods, and products. Experience unrivaled selection, trusted sellers, and seamless transactions. Join a vibrant

community dedicated to animal care and find everything you need in one place. Sulala: Your ultimate destination for all things animal-related in the Middle East.

Key Features

- Unmatched selection: Explore an extensive range of animals, equipment, tools, foods, and products.
- Trusted sellers: Connect with reputable sellers who offer high-quality items.
- Seamless transactions: Enjoy a smooth and secure buying and selling experience.
- Vibrant community: Join a passionate community of animal enthusiasts and professionals.
- Convenience: Find everything you need in one place, saving you time and effort.
- Dedicated to animal care: Access a platform focused on providing the best care for animals

Project Goals

1. **Streamline Animal Management**
    - Provide comprehensive tools for tracking animal data, including breed, age, health status, and medical treatments.
    - Integrate with veterinary care records for seamless health monitoring and appointment scheduling.
2. **Enhance Marketplace Operations**
    - Facilitate efficient buying and selling of animals, equipment, tools, foods, and products.
    - Ensure secure transactions with robust payment options and order tracking.
3. **Empower Vendors**
    - Offer a powerful vendor dashboard for managing product listings, inventory, and sales analytics.
    - Enable easy communication with customers for better service and satisfaction.
4. **Optimize Delivery Logistics**
    - Provide a dedicated delivery partner app for real-time order tracking and optimized route planning.
    - Ensure timely and accurate deliveries with direct customer communication.
5. **Foster a Trusted Community**
    - Build a transparent and reliable platform with verified sellers and user reviews.
    - Promote a community of animal enthusiasts and professionals through interactive features and forums.

Experience the excellence of Sulala and elevate your animal-related endeavors in the Middle East today!

(back to top)

Architecture

The architecture of the Sulala dashboard frontend is designed to provide a robust, scalable, and maintainable solution for managing a comprehensive marketplace platform. It leverages a combination of

modern frameworks, libraries, and tools to ensure a seamless and efficient user experience. The following sections describe the key technologies used in the project, their roles, and how they are utilized.

Technology Stack

1. **Next.js**
    - **Role:** Next.js is a React framework for server-side rendering and static site generation.
    - **Usage:** We use Next.js to handle routing, server-side rendering, and static generation of pages to improve performance and SEO. It helps streamline the development of the frontend with built-in features and optimizations.
2. **NextAuth**
    - **Role:** NextAuth is a library for handling authentication and authorization in Next.js applications.
    - **Usage:** We use NextAuth to manage user sign-in, sign-out, and session management. It supports various authentication providers, which allows us to offer flexible login options and secure access to user accounts.
3. **React**
    - **Role:** React is a JavaScript library for building user interfaces based on components.
    - **Usage:** React is used to create interactive and dynamic UIs. Its component-based architecture helps in building reusable components and managing state and lifecycle in a structured manner.
4. **Tailwind CSS**
    - **Role:** Tailwind CSS is a utility-first CSS framework for styling applications.
    - **Usage:** We use Tailwind CSS for rapid UI development, providing a consistent design system with utility classes. It allows for flexible and responsive design without writing custom CSS.
5. **DaisyUI**
    - **Role:** DaisyUI is a component library built on top of Tailwind CSS.
    - **Usage:** DaisyUI provides pre-built components that enhance development speed and consistency. We use it to integrate ready-to-use UI elements and components, which align with our design requirements.

Libraries and Tools

1. **ChartJS**
    - **Role:** ChartJS is a library for creating interactive charts and graphs.
    - **Usage:** We use ChartJS to visualize data through various types of charts, such as bar charts and line charts. It helps in presenting analytics and metrics in an interactive and engaging manner.
2. **Framer-motion**
    - **Role:** Framer-motion is a library for animations in React applications.

- **Usage:** We use Framer-motion to create smooth and dynamic animations for UI components. It helps enhance user experience by providing visually appealing transitions and interactions.

3. **lodash**

   - **Role:** Lodash is a utility library that provides helpful functions for working with arrays, objects, and functions.
   - **Usage:** We use lodash to simplify common programming tasks such as data manipulation and function handling. Its utilities help in writing cleaner and more efficient code.

4. **husky**

   - **Role:** Husky manages Git hooks to automate tasks before commits and pushes.
   - **Usage:** We use husky to enforce code quality by running linting, formatting, and tests automatically during the commit process. It ensures code consistency and reduces manual errors.

5. **dayjs**

   - **Role:** Dayjs is a lightweight library for date and time manipulation.
   - **Usage:** We use dayjs for handling and formatting dates and times. It provides a simple API for parsing, formatting, and manipulating date values, making it easy to manage date-related operations.

6. **intl-tel-input**

   - **Role:** intl-tel-input is a library for managing international telephone input fields.
   - **Usage:** We use intl-tel-input to provide users with a phone number input field that includes country code selection. It helps in standardizing phone number entry and validation across different regions.

7. **libphonenumber-js**

   - **Role:** libphonenumber-js is a library for validating and formatting phone numbers.
   - **Usage:** We use libphonenumber-js to ensure accurate phone number formatting and validation. It helps in processing and validating phone numbers according to international standards.

8. **react-qr-code**

   - **Role:** react-qr-code is a library for generating QR codes within React components.
   - **Usage:** We use react-qr-code to create QR codes for various purposes, such as user authentication or data sharing. It integrates seamlessly with React to provide QR code generation functionality.

9. **zod**

   - **Role:** Zod is a schema validation library for TypeScript and JavaScript.
   - **Usage:** We use Zod to validate user input and define expected data schemas. It helps in ensuring that data conforms to defined structures and provides clear error messages when validation fails.

10. **zustand**
    - **Role:** Zustand is a state management library for React applications.
    - **Usage:** We use Zustand to manage global state across the application. It provides a simple API for state management and helps in maintaining a consistent application state.
11. **Sentry**
    - **Role:** Sentry is an error tracking and performance monitoring tool.
    - **Usage:** We use Sentry to monitor and track errors, performance issues, and application health. It provides real-time insights and helps in identifying and resolving issues quickly.
12. **Docker**
    - **Role:** Docker is a containerization platform for creating and managing consistent development and production environments.
    - **Usage:** We use Docker to containerize the application, ensuring that it runs consistently across different environments. It helps in simplifying deployment and managing dependencies.

(back to top)

Getting Started

This is a comprehensive guild on how to get started on this project.

Prerequisites

Make sure you have node and npm installed on your system. Node version `v20.12.2` and NPM version `10.5.0`

- npm

```
npm install npm@latest -g
```

Installation

Below is an instruction of installing and setting up sulala app.

1. Setup git with token credential
2. Clone the repo

```
git clone https://github.com/DevSulala/Sulala-ShopDashboard.git
```

3. Install NPM packages

```
npm install
```

4. Configure your environmental variables (in `.env`) using `.env.example`

```
# backend api url
BACKEND_BASE_URL = 'http://<url>/'
# the deployed frontend url of this website
FRONTEND_BASE_URL = 'http://<url>/'
# the url for images bucket
IMAGE_BASE_URL = 'http://<url>/'

# sentry dns
SENTRY_DNS = 'https://<api-key>.ingest.us.sentry.io/<secret-key>'
# sentry auth token
SENTRY_AUTH_TOKEN = 'sntrys_<auth-token>'
# sentry organization id
SENTRY_ORG = '<sentry-org>'
# sentry project id
SENTRY_PROJECT = '<sentry-project>'

# google id for google signup
GOOGLE_ID='<google-id>.apps.googleusercontent.com'
# google secret key for google signup
GOOGLE_SECRET='GOCSPX-<google-secret>'

# random key for next-auth
NEXTAUTH_SECRET='<next-auth-secret-key>'
# random key for next-auth
SECRET="<secret-key>"
# production url for this website
NEXTAUTH_URL='https://<next-auth-url>/'
# development url for this website
NEXTAUTH_URL_INTERNAL='http://<next-auth-dev-url>/'

# google maps key for location auto complete
NEXT_PUBLIC_GOOGLE_MAPS_KEY='<google-geo-encoding-api-key>'
# default number of items to be displayed
NEXT_PUBLIC_DEFAULT_ITEMS_PER_PAGE='20'
# default localization to redirect user when no locale found
NEXT_PUBLIC_DEFAULT_LOCALE='AR'
# to use sentry for monitoring bugs, errors and web performance
NEXT_PUBLIC_USE_MONITORING='<true | false>'
```

5. Run in development mode

```
npm run dev
```

(back to top)

6. There is also `Dockerfile` and `docker-compose.yml` for quick start

```
npm run docker:build
npm run docker:run
```

(back to top)

Folder Structure

Below is an overview of the folder structure used in this project. It uilizes the nextjs conventional app router folder structure.

```
├── CHANGELOG
├── commitlint.config.cts
├── docker-compose.yml
├── Dockerfile
├── LICENSE.txt
├── messages                        # 🗣️ Translation files like ar.json
├── package.json
├── PROJECT_GUIDELINES.md
├── public                          # 🖼️ Assets and Images
├── README.md
├── scripts                         # 🛠️ Scripts to facilitate development
process
│   ├── docker
│   │   ├── BUILD.sh
│   │   └── ...
│   └── git
│       ├── BUMP_VERSION.sh
│       └── ...
├── sentry.client.config.ts
├── ...
├── src
│   ├── actions
│   │   ├── admin-manage            # 🛠️ Server Action for making API
requests
│   │   │   ├── answer-support-request.ts
│   │   │   ├── ...
│   │   ├── ...
│   │   ├── google                  # 🌍 For Google Geolocation encoding
│   │   │   ├── get-location-geocode.ts
│   │   │   └── get-location-suggestion.ts
│   │   ├── mapper                  # 🔄 For Data Transfer Objects
│   │   │   ├── animal-mapper.ts
│   │   │   ├── ...
│   ├── app
│   │   ├── api                     # 🔐 For Google Authentication and API
requests to Next.js
│   │   │   └── auth
│   │   │       └── [...nextauth]
│   │   │           └── route.ts
│   │   ├── error.tsx               # 🚫 Handling Errors
```

```
│   │   ├── favicon.ico
│   │   ├── global-error.tsx       # 🚫 Handling Global Errors
│   │   ├── globals.css
│   │   ├── [lang]                 # 🌐 For Localization
│   │   │   ├── api                # 🔐 For Google Authentication and API
requests to Next.js
│   │   │   │   └── auth
│   │   │   │       └── [...nextauth]
│   │   │   │           └── route.ts
│   │   │   ├── auth               # 🔑 Authentication related pages for
Vendor and Admin
│   │   │   │   ├── approval
│   │   │   │   │   ├── loading.tsx
│   │   │   │   │   └── page.tsx
│   │   │   │   ├── ...
│   │   │   ├── components          # 🛠 Common Components
│   │   │   │   ├── ErrorDisplay.tsx
│   │   │   │   ├── LandingNavBar.tsx
│   │   │   │   ├── LangSwitch.tsx
│   │   │   │   ├── LogoutSwitch.tsx
│   │   │   │   ├── SelectAccount.tsx
│   │   │   │   ├── ThemeSwitch.tsx
│   │   │   │   └── WebsiteUsageAgreement.tsx
│   │   │   ├── dashboard           # 📊 Dashboard related pages for Vendor
and Admin
│   │   │   │   ├── (admin)         # 🛠 Admin page group (the brace won't
have effect on route URL)
│   │   │   │   │   ├── customer-support
│   │   │   │   │   │   ├── detail
│   │   │   │   │   │   │   ├── page.tsx
│   │   │   │   │   │   │   └── utils
│   │   │   │   │   │   │       └── helper.util.ts
│   │   │   │   │   │   ├── error.tsx
│   │   │   │   │   │   ├── loading.tsx
│   │   │   │   │   │   ├── not-found.tsx
│   │   │   │   │   │   ├── page.tsx
│   │   │   │   │   │   └── schema  # 📚 For Table Schema
│   │   │   │   │   │       ├── data.ts
│   │   │   │   │   │       ├── schema.ts
│   │   │   │   │   │       └── type.ts
│   │   │   │   │   ├── ...
│   │   │   │   │   └── statistics  # 📈 For Statistics
│   │   │   │   │       ├── page.tsx
│   │   │   │   │       ├── ...
│   │   │   │   ├── layout.tsx
│   │   │   │   ├── ...
│   │   │   ├── error.tsx
│   │   │   ├── layout.tsx
│   │   │   ├── not-found.tsx
│   │   │   ├── opengraph-image.tsx # 🖼 For Open Graph Image description
(LinkedIn, Twitter, etc.)
│   │   │   ├── page.tsx
│   │   │   └── ...
│   │   ├── layout.tsx
```

```
│   │   ├── manifest.ts             # 📄 For PWA and SEO Indexing
│   │   ├── not-found.tsx
│   │   ├── page.tsx
│   │   └── sitemap.ts              # 🗺️ For Web Crawlers
│   ├── components                  # 🔧 Common Components for all Pages
│   │   ├── AuthWithEmail.tsx
│   │   ├── AuthWithPhone.tsx
│   │   └── common
│   │       ├── form
│   │       │   ├── ColorPaletteInput.tsx
│   │       │   └── ...
│   │       ├── modal
│   │       │   ├── CropImageModal.tsx
│   │       │   └── ...
│   │       ├── table
│   │       │   ├── TableActions.tsx
│   │       │   └── ...
│   │       └── ui
│   │           ├── BackButton.tsx
│   │           └── ...
│   ├── config
│   │   ├── table.config.ts
│   │   └── urls.ts
│   ├── constants
│   │   ├── countries.json
│   │   └── ...
│   ├── error                       # 🛠️ Custom Errors
│   │   └── custom-zod.error.ts
│   ├── hooks
│   │   ├── useCreateQueryString.ts
│   │   └── ...
│   ├── i18n                        # 🌐 Localization Config
│   │   ├── config.ts
│   │   └── navigation.ts
│   ├── i18n.ts
│   ├── instrumentation.ts          # 📊 For Instrumentation and Monitoring
with Sentry
│   ├── lib                         # 🛠️ Common Libraries for all Pages
│   │   ├── detect
│   │   │   ├── client.ts
│   │   │   └── server.ts
│   │   └── ...
│   ├── middleware                  # 🛢️ Middleware
│   │   ├── authMiddleware.ts
│   │   └── ...
│   ├── middleware.ts
│   ├── providers                   # 🛠️ Zustand Store Providers
│   │   └── setup-account-store-provider.tsx
│   ├── stores
│   ├── types
│   │   ├── props.type.ts
│   │   └── ...
│   └── utils
│       ├── convertDataURLtoFile.ts
```

```
│       └── ...
├── ...
└── tailwind.config.ts
```

This Project utilizes NextJs.

1. **SSR (Server Side Rendering)**:

- Mainly for SEO Optimization and protecting some data fetching endpoints like google map key and reduce request and extra calculation on the users browser.

2. **SSG (Server Side Generation)**:

- The landing page is made to be SSG to reduce server side rendering on each request and since there is no data fetching needed for the landing page.

3. **CSR (Client Side Rendering)**:

- Most of Dynamic Forms and Input uses CSR for interactivity with the user

4. **SA (Server Actions)**:

- All of data request to the backend are made with the new nextjs server actions. This enable us to request data using Fetch and also stateful response using FormState. Beside this, it hides the implementation and url, enabling us to secure google maps key and other things.

SEO Optimization

This website is built to be friendly with SEO engines. Below is list of optimization techniques this project utilizes.

1. **Manifest File**: is a simple file that tells the users browser about the website, and how it should be installed on the users mobile or desktop device. This makes it possible that this website to be installed on users mobile application as PWA website

2. **OpenGraph**: this allows us to set Open Graph and Twitter images for a route segment. They are useful for setting the images that appear on social networks and messaging apps when a user shares a link to sulala site.

3. **SiteMaps**: is a file that contains information about this site's pages, images, and other files and the relationships between them. Search engines like Google read this file to more intelligently crawl your site.

4. **Robots File**: is a file that tells search engine crawlers or bots which URLs they can access or not on your website. This help us to prevent search engines from indexing certain private pages

5. **MetaTags(title and description)**: this defines the application meta data dynamically (in arabic or in english dynamically) and help search enginer to index the page

6. **Canonical Tags**: this helps search engines understand which version of website page should be preferred version when multiple pages have similar or identical content. Since this website have only

one version, the canonical url might not have usecase, but it is implemented incase this website going to have multiple versions in the future.

For more examples, please refer to the Documentation

(back to top)

Usage

Running the app and going to the browser `localhost:3000`, you will see the landing page .

For more examples, please refer to the Documentation

(back to top)

Helpers

This project also utilize some helper `scripts` and `git hooks`.

1. Scripts:

- the `scripts` folder holds two types of scripts, for docker, and for git.
- the git scripts are used for tagging commits and pushing the tag to source code repository
- the docker scripts are used for building docker images and pushing it to docker registry
- the script increament the version based on args given.
- argument `--patch` or no argument increases the patch version of the app.
- argument `--minor` increases the minor version of the app.
- argument `--major` inceases the major version of the app.
- *semantic versioning* strategy is used in this case. to create tag or version, use the following guidelines.
    - **patch**: Apply a patch for bug fixes, style improvements, and other minor changes.
    - **minor**: Apply a minor version for changes that go beyond bug fixes, including the addition of new features, without breaking existing functionality.
    - **major**: Apply a major version for features that introduce significant changes and may cause compatibility issues with previous versions of the app, potentially leading to crashes.

1. Hooks:

- This project utilizes Git hooks to perform various operations on Git actions, such as:
    - Linting and formatting code using Prettier, and checking commit messages before committing changes (`pre-commit`).
    - Standardizing commit messages (`commit-msg`).
    - Running tests before pushing changes (`pre-push`).
    - Automatically tagging code and installing packages after merging and pulling changes (`post-merge`).

For more examples, please refer to the Documentation

(back to top)

Roadmap

- [x] Add Changelog
- [x] Add back to top links
- [x] Add meaningful commit message (eg `fix: fix issue with something`). Other wise husky will not let you continue.
- [x] Multi-language Support
  - [x] Arabic
  - [x] English
- [x] Theme Support
  - [x] Dark Mode
  - [x] Light Mode
- [ ] Add Additional Templates w/ Examples
- [ ] Add "components" document to easily copy & paste sections of the readme

For more examples, please refer to the CHANGELOG

See the open issues for a full list of proposed features (and known issues).

(back to top)

Contributing

Started working on this project? Please follow the following guidline to get started on working on this project

1. Fork the Project
2. Create your Feature Branch (`git checkout -b feature/AmazingFeature`)
3. Commit your Changes (`git commit -m 'feat: Add some AmazingFeature'`)

> when commiting message, use the git commit message conventions by prefixing you commit message with `feat`, `fix`, `docs`, `chore`, `style`, `refactor`, `ci`, `test`, `revert`, `perf`, `vercel` eg.

```
git commit -m 'feat: added something'
```

4. Push to the Branch (`git push origin feature/AmazingFeature`)
5. Open a Pull Request

(back to top)

License

Distributed under the MIT License. See `LICENSE.txt` for more information.

(back to top)

Contact

Sulala - @sulala_twitter - email@sulala.com

Project Link: https://github.com/DevSulala/Sulala-ShopDashboard

sulala.com

(back to top)