Github Link: https://github.com/dvrshi/akasa-food
Deployment Link: https://akasa-food.vercel.app/
Demo Video: Akasa Foods.mp4
Note: While using the hosted website, there may be a little delay experienced in loading cart history and order history or while signing in and out, kindly ignore it.
Frameworks/Tech Stack used:

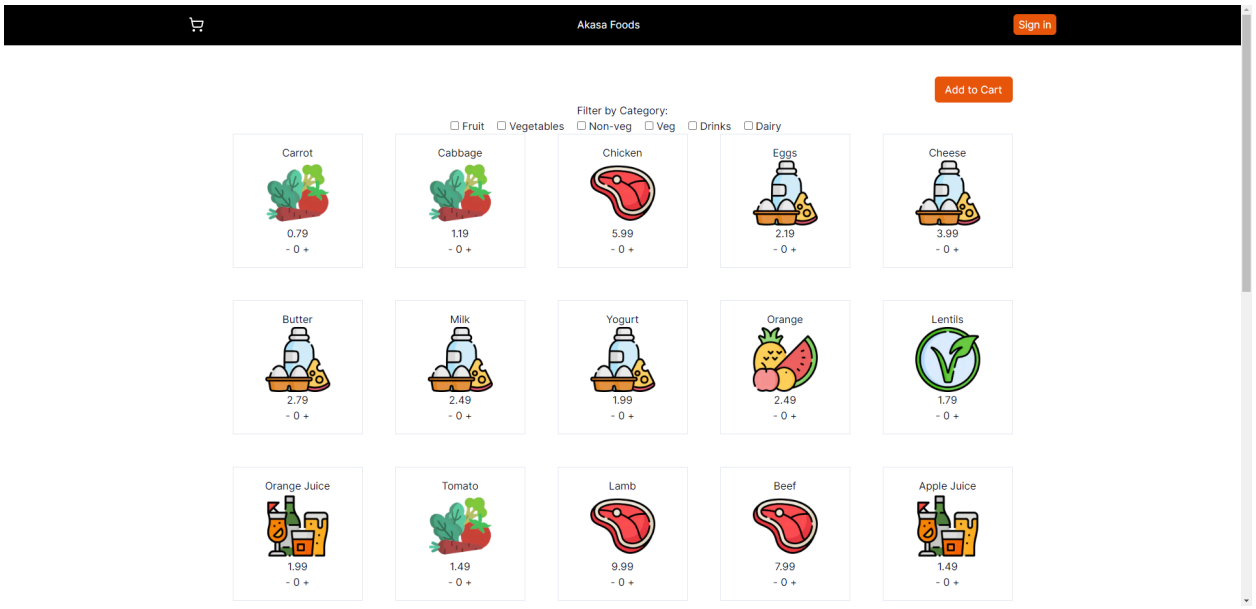
- NextJS and Vercel for website deployment
- TailwindCSS for styling
- Prisma for connecting PostgreSQL to the webapp
- Supabase for accessing the database
- NextAuth for user authentication
- Shadcn for UI components
- Zod for implementing user registration validation

Steps to install the code base:

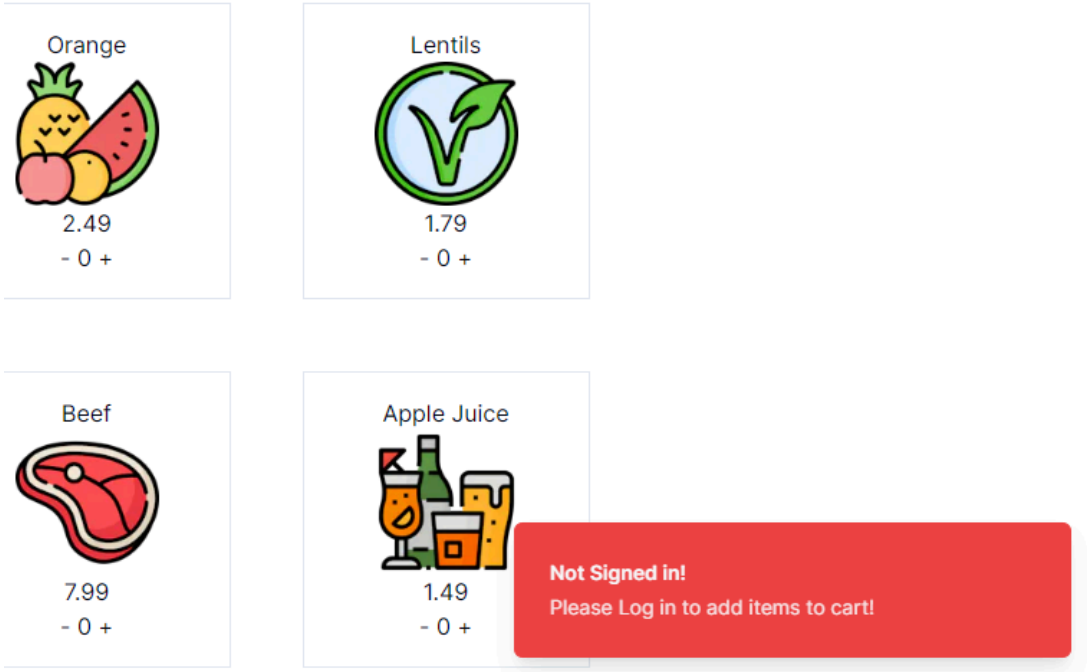
1. Download the github repository from the link, and run ‘npm install’ in the terminal to install all the required libraries.
2. Then, use the command ‘npm run dev’ to open up the website using localhost.

Website Details:

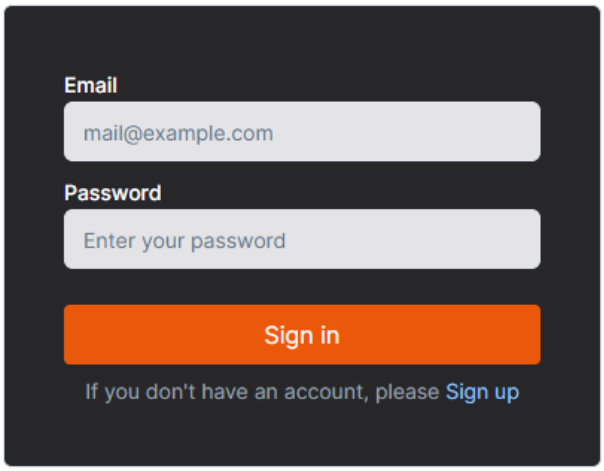
Homepage:

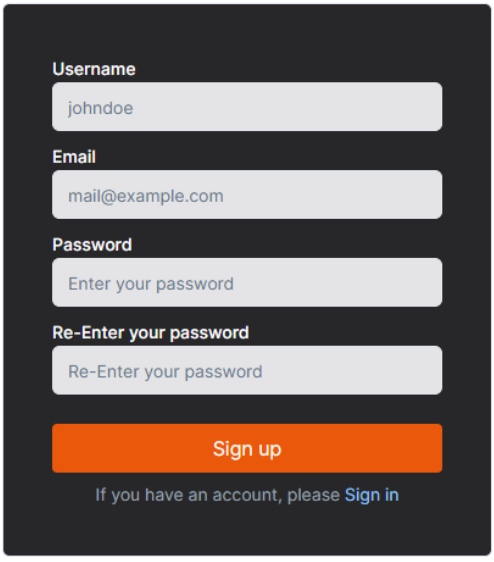


- This is the default homepage for the website, although it shows the menu and add to cart option you would need to log in first to save the cart.



- Session functionality has been implemented so that there can be a track of the user and then only can the cart be saved after successful login.



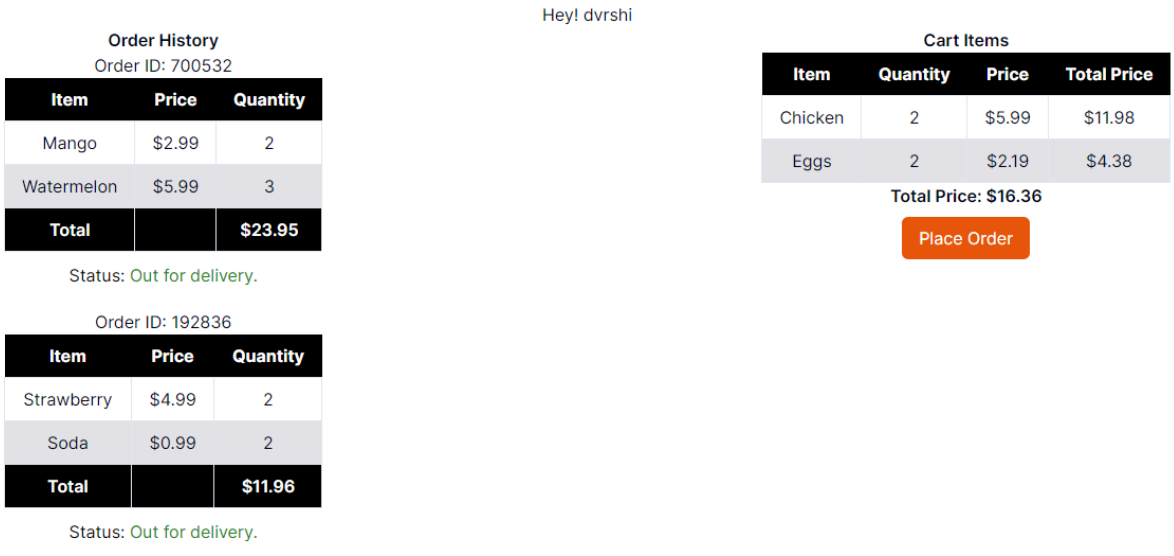


- Separate sign in and sign up pages have been created. I have used zod to check if email id is valid, the password matches the required length and if the password and confirm password fields match.

The database schema is as follows:

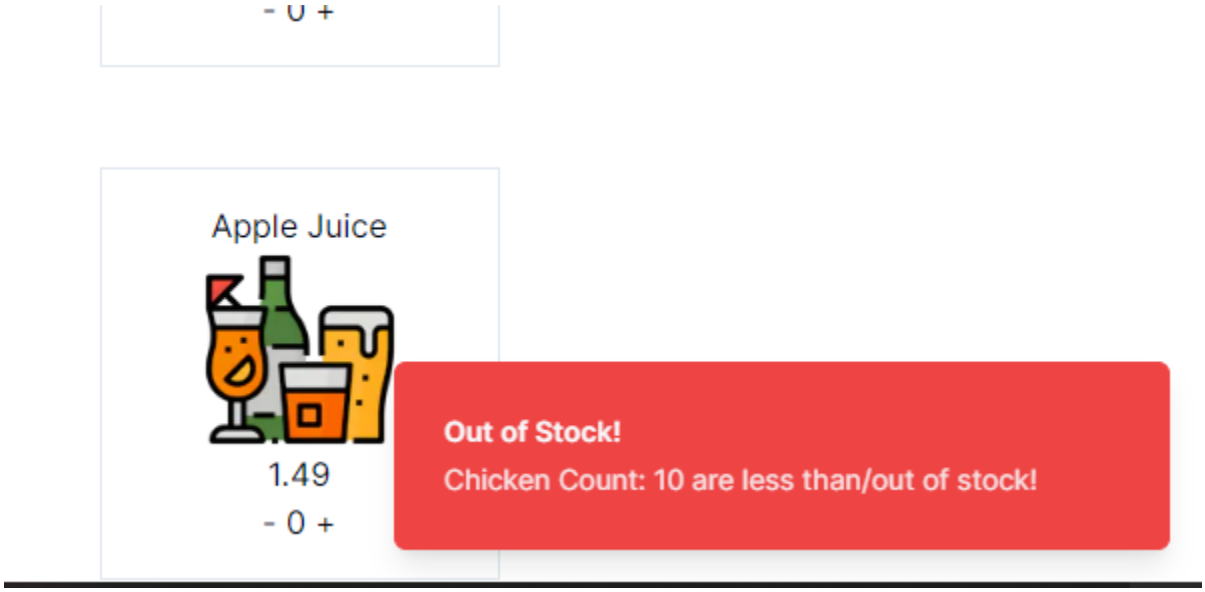
```
model User {
  id       Int      @id @default(autoincrement())
  email    String   @unique
  username String   @unique
  password String
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
  cart     Json
  orders   Json[]  @default([])
}

model Items {
  id            Int       @id @default(autoincrement())
  name          String
  category      String
  price         Float
  stockQuantity Int
}
```

There are two tables created, first to store the user details and second to store the inventory.

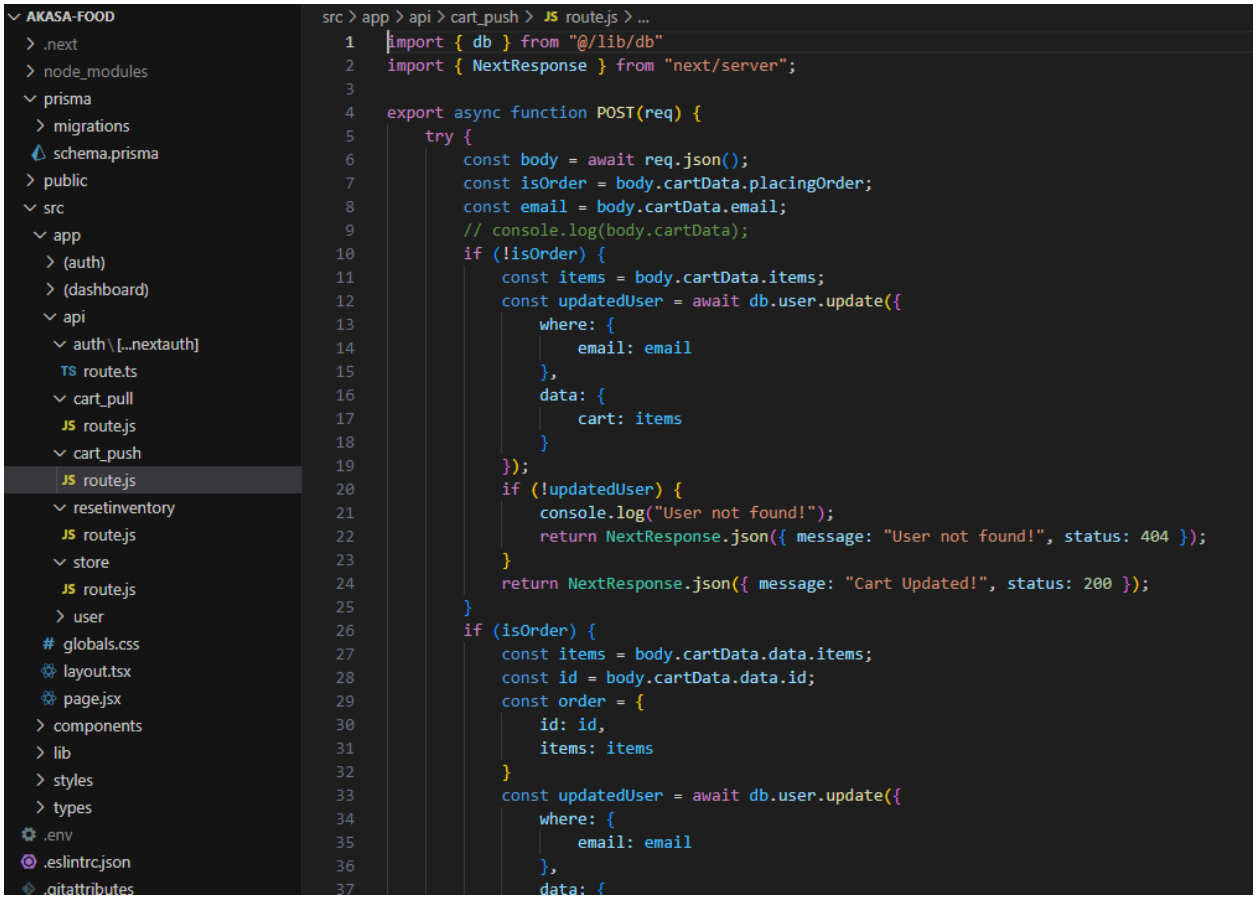


We can see the current cart and order history on the website by clicking on the shopping cart icon in the navigation bar.

- All edge cases have been taken care of, users cannot order any items that are out of stock.
- We can log in from multiple devices.
- We can also save the cart from one device and view cart/place order on another device. Which was possible through session management.
- If between the time a user saves the cart and places the order, an item goes out of stock, then also the user is notified of the item and if it is there in the inventory in limited quantities.
- If two users order at almost the same time, if the inventory then becomes low or empty after one of them orders, then the other user will not be able to place the order.
- The filter buttons also work seamlessly, while browsing the menu page.
- There has also been a separate feature provided to restock the inventory database, use the email 'admin@gmai.com' and the password 'asdfasdf' to login. Then click on the shopping cart button and click the restock button. This will send an api call to the database and thus restocking the inventory.
- Incase first time visiting the website, click in SignIn button and then open the Signup form to create an account.
- Email and username will be unique to all users and also a basic password strength checker has been implemented.
- Checks have been implemented so that users do not order out of stock items.



- There are separate API calls for updating the user table, for user creation, saving cart history, reading cart history, saving and reading order history. These can be seen inside the ‘api’ folder inside the source code directory.



- Through Implementation of session tokens, the unauthorized usage has been prevented

```jsx

import { authOptions } from '@/lib/auth';
import { getServerSession } from 'next-auth';
import LoadCartData from '@/app/(dashboard)/user/LoadCartData.jsx';
import LoadOrderHistory from '@/app/(dashboard)/user/LoadOrderHistory.jsx';
import ResetInventory from '@/app/(dashboard)/user/ResetInventory.jsx';
async function page() {

    const session = await getServerSession(authOptions);
    // console.log(session);
    if (session?.user.username === 'admin')
        return (
            <div>
                <ResetInventory session={session} />
            </div>
        );
    if (session?.user)
        return (
            <div className='w-full text-center'>
                Hey! {session?.user.username}
                <div className='flex justify-between'>
                    <div style={{ marginLeft: '25rem' }}>
                        <LoadOrderHistory session={session} />
                    </div>
                    <div style={{ marginRight: '25rem' }}>
                        <LoadCartData session={session} />
                    </div>
                </div>
            </div>


        )
    return (
        <div>Please login</div>
    );
}

export default page;

```

Please look at the source code for specific implementation details.