Frontend Intern Test
T.H.R Chathuri Nirosha

Project Documentation

Design Decisions Explanation

For "Delicious Bites," I wanted the design to feel premium and immersive, not just functional. I chose a **glass morphism style** with blurs and transparency because it makes the menu feel like a modern app you'd want to touch. It wasn't just about looking good; it helped separate the food images from the background, making the dishes the real stars of the show.

I also decided to **remove the manual theme toggle**. Honestly, most people just want their apps to match their phone's settings. By syncing automatically with the system's dark or light mode, the app feels more integrated and "smart" without forcing the user to find a settings button.

Since this is a menu, I assumed most people would view it on their phones. That's why I took a **mobile-first approach**. I made sure buttons were big enough for thumbs (48px+), and I used horizontal swipes for categories to save vertical space. Adding **Framer Motion** was the final touch the little bounces and smooth transitions make the app feel fast and responsive, masking any tiny delays when fetching data.

Technical Challenges Faced

1. Keeping the URL and UI in Sync

One tricky part was making sure the search bar and the URL always told the same story without making the app feel slow. If I updated the URL on every keystroke, it lagged. If I didn't update it at all, you couldn't share a link to a specific search. I fixed this by using a "debounce" time basically waiting for the user to stop typing before changing the URL, while updating the UI instantly so it felt snappy.

2. The Dark Mode "Flash"

Getting the automatic dark mode working with Next.js was annoying at first. Because the server renders the page before knowing the user's preference, there was often a split-second flash of the wrong theme. I solved this by building a custom `ThemeProvider` that handles the check immediately, preventing that jarring flicker when the page loads.

3. Making the Mock API Safe

Since I was using `json-server`, I couldn't be 100% sure the data would always look perfect. I didn't want the app to crash if a price was missing or a list was empty. To fix this, I wrote strict TypeScript interfaces and added safety checks (like default values) in the API calls. Now, even if the backend returns something unexpected, the app stays running smoothly.

Contact Number - +94 741633304
Gmail – chathurinirosha112233@gmail.com