LiveHindustan Front Page Clone –

Technical Submission Overview:

This assignment implements a simplified front-page clone of LiveHindustan using Next.js, TailwindCSS, dynamic routing, SEO metadata, mock data + live NewsAPI, and optimized images.

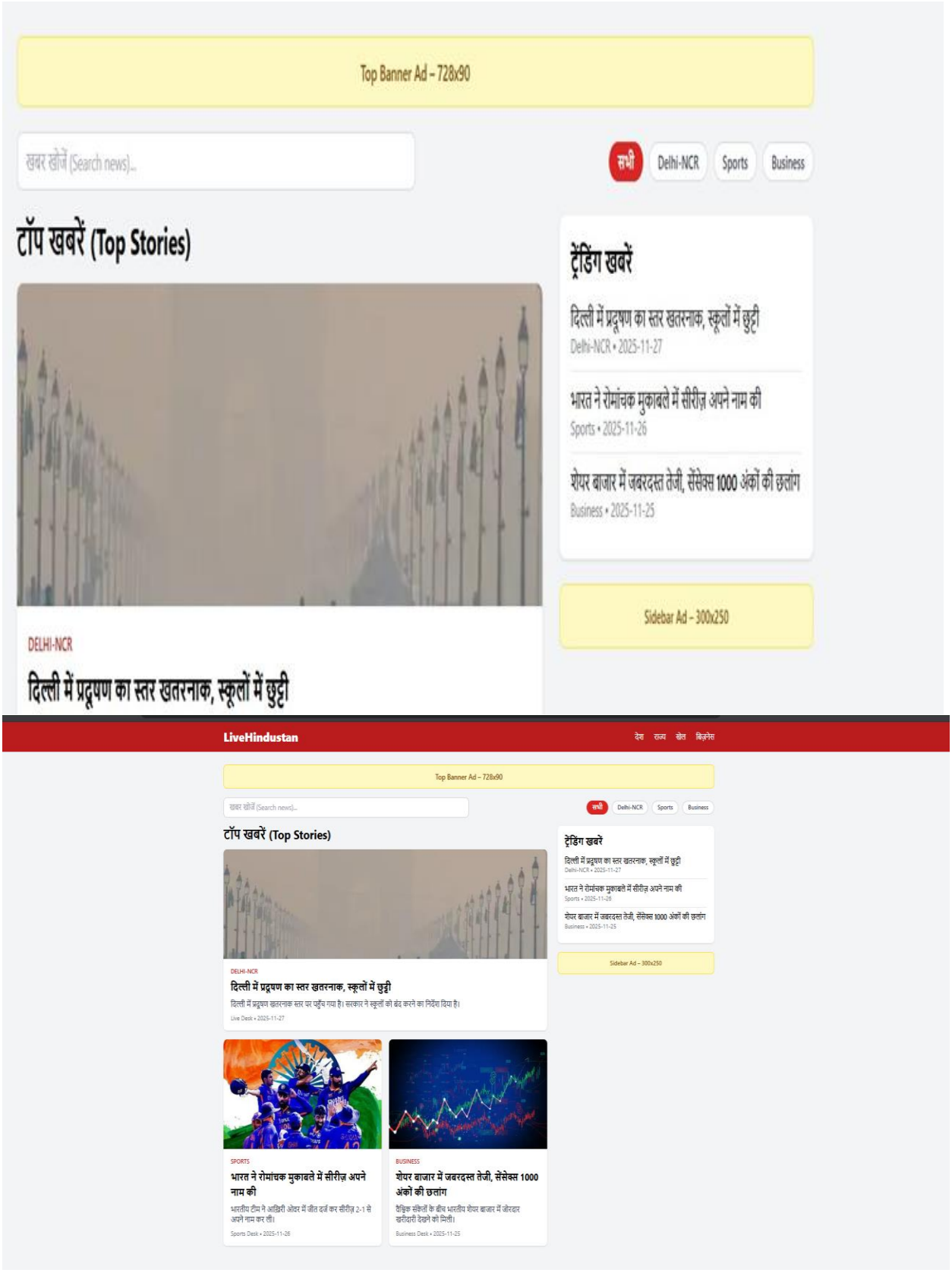
Key Features:

- Responsive UI with hero story, grid layout, and right-side trending section.
- Dynamic routing for article pages using /news/[slug].
- Data fetching via getServerSideProps (NewsAPI) and getStaticProps (local mock data).
- Image optimization using next/image.
- Search + category filtering.
- Advertisement placeholders (top banner, sidebar).
- SEO using + JSON-LD structured data.

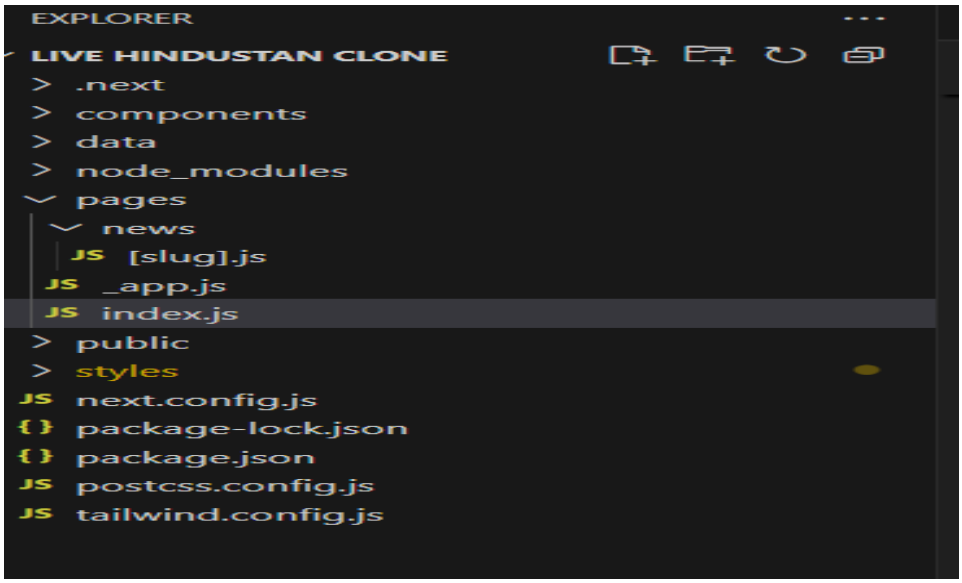
Architecture:

- Next.js Pages Router
- components/: Layout, Header, NewsCard, Ads
- data/news.js: mock API
- public/images/: optimized static assets

Final UI Screenshot:



Project Folder Structure:



Data Model (Brief):

Each article includes: { id, slug, title, category, imageUrl, summary,

content, author, date }. Testing & Edge Cases:

- API failure → API block hidden gracefully.
- Missing images → fallback placeholder supported.
- Long titles wrap naturally.

(ADS are not placed because it needs HTTPS AND needs to be AdSense approved since the assignment was about using only react)

Empty search → friendly ‘no results’ message.

Testing / Edge Cases:

1.What happens if an article doesn’t have an image?

In **NewsCard.js** and [slug].js:
CODE-const imageSrc = article.imageUrl || "/images/placeholder.jpg";

Use **imageSrc** instead of **article.imageUrl:**
CODE-<Image
  src={imageSrc}
  alt={article.title}
  fill
  className="object-cover"
/>

If imageUrl is missing in the article object, I fall back to a default /images/placeholder.jpg. This keeps the layout stable and avoids broken images.

2.No news available” / empty state

If search + category filters result in no matching articles, I show a friendly ‘no results’ message instead of an empty grid. This covers the case where local data is empty for a given filter.

If getAllArticles() ever returns an empty list, the hero card will be skipped and only the empty-state message will show, so the layout still renders without crashing.

3.Loading state while data is being fetched

The home page uses getServerSideProps, so the user only sees the page after data is loaded on the server. There is no visible client-side loading spinner because the HTML is sent with data already filled in. If I later move API calls to the client with fetch/SWR, I would add a loading indicator and reuse the same empty-state message when no news is available.

4.How does layout handle long titles?

Long titles wrap across multiple lines because of normal CSS text wrapping. I use leading-snug and responsive font sizes so even long Hindi titles remain readable without breaking the card layout.

```
<h2 className="font-bold text-base md:text-lg leading-snug">
  {article.title}
</h2>
```

AI Use + Reflection

I used AI support selectively to speed up development, mainly for:

- Generating initial component boilerplates (Layout, Header, NewsCard, Ad components).
- Helping structure Next.js data-fetching logic using getServerSideProps.
- Writing TailwindCSS utility classes for layout spacing, responsiveness, and typography.
- Creating dynamic routing setup for /news/[slug].js.
- Drafting SEO meta tags and structured data schema.
- Providing fallback handling patterns (no image, no results, error boundaries).

Where AI suggestions were wrong or suboptimal

During development, some AI-generated suggestions were not directly usable:

- Incorrect import paths and mismatched component names that didn't match my folder structure.
- AI suggested SWR client-side fetching even though the assignment

required SSR/SSG.

- I corrected this by switching to getServerSideProps.

- Some Tailwind styles (like fixed widths/heights) broke responsiveness.
- I replaced them with flexible grid and percentage-based layouts.

How I verified AI-generated code

- Testing imports / component usage directly in VS Code, checking errors in terminal.
- Running the Next.js dev server (npm run dev) to verify hydration, routing, and console logs.
- Manually adjusting layout and CSS after observing real behaviour in the browser.

Custom modifications I made beyond AI suggestions

- Added category filtering, search functionality, and empty-state messages manually.
- Integrated fallback logic for missing images, long titles, and empty API responses.
- Added multiple ad components (top banner, sidebar ad, inline box ad) not suggested by AI.
- Customized site-wide layout to replicate LiveHindustan’s look instead of generic news layout.