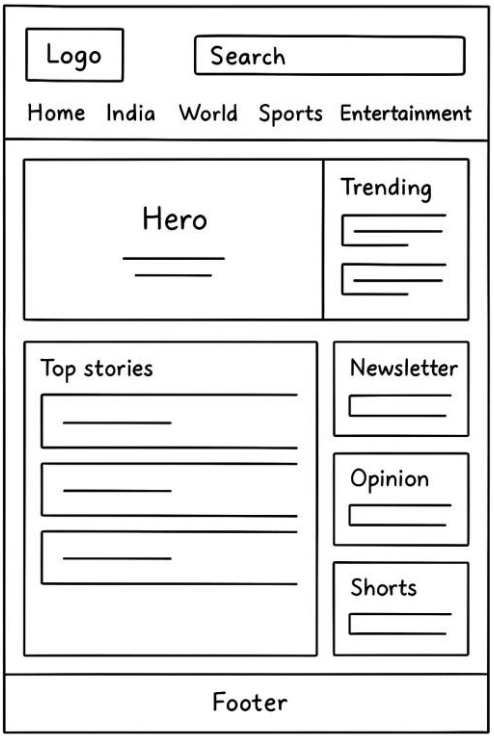
1. Design Document



2. Layout Decisions

- I used a **two-column layout** (wide left column + narrow right) because news websites prioritise readability and highlight secondary widgets (Trending, Newsletter) in the sidebar.
- The **Hero section** is kept large at the top because users first look for the biggest story of the day.
- **Card-based design** ensures reusable UI for articles across categories.
- The site becomes **single column** on mobile for better accessibility and scrolling experience.
- High priority stories use **larger images and bold headlines** to make them visually prominent.

3. Data Fetching Strategy

- I used **getStaticProps (Static Site Generation) with Incremental Static Regeneration** to fetch articles.
- **Reasoning:**
    - Fast load time
    - SEO-friendly
    - News updates every few minutes without redeploying

- **Trade-offs:**
    - Slight delay in showing the *freshest* news due to revalidation timer.
    - Client-side fetch is more real-time but weaker for SEO.

4. Code Explanation

Components Created:

- **Header / Navbar** – Logo, navigation links, search
- **Hero** – Large top story with image + title
- **ArticleCard** – Used for all article previews
- **Trending Sidebar** – Trending/Most Read section
- **Newsletter Box** – Email input + CTA
- **Footer** – Basic footer links
- **Layout Wrapper** – Common container for every page

Data Model (Article Object):

```
{
  "title": "Sample Headline",
  "image": "https://...",
  "summary": "Short summary",
  "date": "2025-11-29",
  "category": "India"
}
```

Challenges Faced:

- Missing global CSS path → fixed by correcting import and restarting dev server.
- News API returning inconsistent data → added fallback image and placeholder summary.
- Responsive problem on mobile → added flex + grid breakpoints.
- AI-generated code sometimes outdated (older Next.js functions) → manually corrected imports.

If Given More Time, I Would Improve:

- Add infinite scroll
- Add search suggestions

- Add category-based dynamic pages
- Improve accessibility (WCAG compliance)
- Add dark mode toggle

PART C – TESTING / EDGE CASES

1. Missing Image

Expected behavior:
Show a placeholder image.

if (!article.image) show "placeholder.jpg"

2. API Returns No Articles

UI should show:

"No news available right now. Please try again."

[Retry Button]

3. Very Long Titles

Handled by using:

line-clamp: 2;

text-overflow: ellipsis;

4. Loading State

Before articles load, show skeleton cards.

5. Error State (Fetch Failure)

Show:

"Unable to load news. Check your connection."

PART D – AI USE + REFLECTION

How AI Helped in the Project

- Generated component boilerplate for Hero, Navbar, and Cards
- Suggested CSS layout structure
- Helped write data fetching logic example
- Assisted in documentation formatting

Where AI Was Wrong

- Provided outdated Next.js file structure (pages/ instead of app/)

- Suggested incorrect imports (getInitialProps)
- Overcomplicated CSS at some places → manually simplified
- Misplaced globals.css path causing build error

How I Verified & Corrected AI Code

- Tested components locally
- Simplified the AI-generated CSS into responsive grid classes
- Replaced incorrect Next.js methods with getStaticProps
- Ensured TypeScript/JS runs error-free by linting and running build

Custom Work Beyond AI

- Full layout logic & component structuring
- Implementing fallback images
- Designing loading skeletons
- Debugging build errors manually
- Improving responsiveness using real device testing