Fitpeo®

Web Developer Assignment: "Healthcare Dashboard (React - Static Display)"

Referenced Design Image:
https://i.ibb.co/B2DdGkgF/Screenshot-2025-05-23-at-3-31-31-PM.png

---

Objective:

Implement a single-page "Healthcare Dashboard" using **React**, focusing on accurately recreating the visual design from the provided image. The primary goal is to demonstrate your ability to:

1. **Break down a complex UI into reusable React components.**
2. **Manage and display static data** from mock JavaScript sources via props.
3. **Apply styling for pixel-perfect visual fidelity** to the provided design image.
4. **Ensure full responsiveness** across different screen sizes.

Important Note: No interactive functionality (clicks, forms, filtering, adding, editing, deleting) is required for this assignment. The dashboard should be a visually faithful, static representation of the design, rendering all data from mock sources.

Design Concept:

Your task is to precisely recreate the clean, functional healthcare dashboard as depicted in the provided image. It features a distinct header, a fixed navigation sidebar, and a dynamic main content area displaying static health data, an anatomical illustration with associated info, and an appointment scheduling overview. The design clearly incorporates calendar views and activity charts.

Visual Elements to Adhere To (Directly from the image):

- **Overall Theme:** Minimalist, professional, clean, and organized.
- **Color Palette:** Observe and replicate the shades of light blues, grays, and white used in the header, sidebar, and background, along with specific accent colors for icons, status indicators, and highlights (e.g., green for 'Healthy Heart', red for 'Lungs' status, specific calendar blues).
- **Typography:** Identify and use fonts that closely match the style and weights seen in the image for titles, body text, and numerical data. Pay attention to font sizes and line spacing.
- **Icons:** Replicate the specific icons used in the sidebar navigation, search bar, user profile, anatomical sections, and appointment cards. You can use icon libraries (like Font Awesome or Lucide React) to find close matches or use SVG/image assets.
- **Imagery:** Accurately incorporate the anatomical illustration (human body) and the specific health indicators (Healthy Heart, Lungs, Teeth, Bone) as shown. Use open-source images or similar illustrations if you cannot extract them directly. Any Placeholder image also works.
- **Layout & Spacing:** Meticulously recreate the exact spacing, padding, margins, and alignment of all elements as seen in the image. This includes the width of the sidebar, the layout of the main content grid, and the internal spacing of cards.
- **Shadows/Borders:** Replicate the subtle shadows and border radii on cards, search bars, and other UI elements to match the depth and softness of the original design.

Section Breakdown (To be implemented as React Components):

You should break down the UI into logical, reusable React components to reflect the structure observed in the image.

1. `App` **Component (Root Component):**
    - Will orchestrate the main layout (e.g., using Flexbox or Grid for `Header`, `Sidebar`, and `DashboardMainContent`).

2. `Header` **Component (Top Bar):**
    - **Content:**
        - App Logo/Title (`Healthcare.`).
        - Search Bar (display only, no functionality).
        - Notification Icon.
        - User Profile (static avatar and name).
        - "Add" Button (e.g., `+` icon, display only).
3. `Sidebar` **Component (Left Navigation):**
    - **Content:**
        - "General" heading.
        - Navigation Links (Dashboard, History, Calendar, Appointments, Statistics, Tests, Chat, Support, Setting – all as static links, no active highlighting needed beyond initial render).
4. `DashboardMainContent` **Component (Main Area Container):**
    - This component will contain and render the various sub-sections of the dashboard, as shown in the image.
    - **Sub-components within `DashboardMainContent`:**
        - `DashboardOverview` **Component:**
            - `AnatomySection` **Component:**
                - Displays the anatomical illustration (human body) with static indicators and text (e.g., "Healthy Heart", "Lungs", "Teeth", "Bone") positioned as in the image.
                - Each indicator should have its associated health status (e.g., green for healthy, red for an issue).
            - `HealthStatusCards` **Component:**
                - Displays static cards for "Lungs", "Teeth", "Bone" with mock dates and status indicators, as positioned next to the anatomical figure.
        - `CalendarView` **Component:**
            - Displays a static monthly calendar grid (e.g., "October 2021").
            - Statically render specific appointment times on relevant days (e.g., "09:00", "11:00", "13:00", "15:00") as shown.

        - Below the main calendar, display static appointment details for "Dentist" and "Physiotherapy Appointment" cards, including names and times.
    - **`UpcomingSchedule` Component:**
        - Displays the static "The Upcoming Schedule" section.
        - Organized by day (e.g., "On Thursday", "On Saturday").
        - Contains multiple instances of the `SimpleAppointmentCard` component.
    - **`ActivityFeed` Component:**
        - Displays the static "Activity" section, including the "3 appointments on this week" text and the bar chart visually. This chart can be a static CSS representation of bars, no actual data plotting is needed.
5. **`SimpleAppointmentCard` Component:**

    - A reusable component for the appointment cards in the "Upcoming Schedule" section.
    - **Content (static):** Title (e.g., "Health checkup complete", "Ophthalmologist", "Cardiologist", "Neurologist"), Time, and a small icon/indicator.

Technical Requirements:

1. **Framework:** React (use `create-react-app` or Vite for project setup).
2. **Component-Based Architecture:**
    - Break down the entire UI into logical, reusable React components as outlined above.
    - Use functional components with Hooks.
3. **Data Handling (Static):**
    - Create **JavaScript arrays of mock data** in separate files (e.g., `src/data/appointments.js`, `src/data/healthData.js`) for:
        - Navigation links.
        - Anatomical health status indicators.
        - Calendar appointment times and details.
        - Upcoming appointments.
    - Pass this mock data down to components using **props** for rendering.
    - **Crucially, no state management for user interaction (e.g., `useState` for toggling visibility based on clicks, `useEffect` for data fetching)**

is required. All data should be pre-defined in your mock data files and rendered directly.

4. **JSX & Styling:**
    - Implement the structure using JSX within your React components.
    - Apply styling using:
        - Standard CSS (e.g., `App.css`, component-specific `.css` files).
        - Or, CSS Modules (e.g., `.module.css`).
        - Or, Styled Components (if preferred).
    - **Responsiveness:** The entire dashboard must adapt gracefully to different screen sizes (desktop, tablet, mobile) using CSS Media Queries, Flexbox, and/or CSS Grid within your React components' styling. **This is a key requirement for visual fidelity.**
    - **Pixel-Perfect Fidelity:** Pay extreme attention to detail in replicating the exact layout, spacing, alignment, typography, colors, shadows, and border-radii as seen in the provided image.
5. **Assets:**
    - Use open-source images or SVG icons for the anatomical illustration, user avatars, and any other visual elements from the design. Ensure all static assets are correctly imported and displayed within React.
6. **File Structure:**
    - Organize your React project logically (e.g., `src/App.js`, `src/index.js`, `src/components/`, `src/data/`, `src/styles/`).
7. **Code Quality:**
    - Write clean, well-organized, and commented code.
    - Follow React best practices for component structure, prop usage, and rendering lists.
    - Ensure proper accessibility attributes where applicable (e.g., alt text for images).

Submission Requirements:

You must complete the following steps:

1. **Develop the Project:** Build the entire React application according to the specifications above.

2. **Initialize Git Repository:** In your project's root directory, initialize a Git repository (`git init`), add all your project files (`git add .`), and make an initial commit (`git commit -m "Initial dashboard setup"`).
3. **Create Remote Git Repository:** Create a **public** repository on a Git hosting service (e.g., GitHub, GitLab, Bitbucket).
4. **Push Code to Remote Repository:** Link your local repository to the remote one and push all your project code. **Ensure your resume is NOT included in this code repository. Add it in a separate main folder with Folder Name: Resume**
5. **Host the Website:** Deploy your React application to a static site hosting provider. Recommended free/easy options include:
    - **Vercel** (highly recommended for React apps)
    - **Netlify**
    - **GitHub Pages** (requires specific setup for React SPAs)
    - **Render.com** (static sites)
    - Follow the chosen provider's documentation to deploy your React app from your Git repository.
6. **Fill Out the Submission Form:** Go to the provided form link: https://forms.cloud.microsoft/r/uQavtsVDa7
    - Provide the **public Git Repository Link** for your project.
    - Provide the **Live Website URL** for your hosted application.

Evaluation Criteria:

- **Visual Fidelity (50%):** How accurately does the implemented dashboard visually match the layout, spacing, colors, typography, and element placement of the provided design image?
- **React Component Structure (20%):** Is the UI effectively broken down into logical, reusable, and well-named React components? Is static data correctly passed down through props?
- **Responsiveness (15%):** Does the dashboard adapt gracefully and maintain its visual integrity across various screen sizes (desktop, tablet, mobile)?
- **Code Quality (15%):** Is the React code clean, well-structured, maintainable, and does it follow React best practices? Are CSS styles organized and effective?

Important Disqualification Notice:

- "**We will contact only if you are selected for the next level**."

- **"Disqualified if AI copied or plagiarized code is there."**

Good luck with your assignment! This is a challenging task that will showcase your frontend development skills.