**HealthVisualizer – the new programmer's overlap doc**

1. **HealthVisualizer** is a Next.js healthcare platform featuring patient, doctor, and instructor interfaces. It provides appointment management, health metrics tracking with BMI calculations and data visualization, and includes a comprehensive authentication system with role-based access.

1. **System Architecture Overview:**

תמונה שמכילה טקסט, צילום מסך, תרשים, עיצוב

התיאור נוצר באופן אוטומטי

1. **Core industry lead Technologies:**
   1. Next.js Framework - next.js is a Modern React framework that offering server-side rendering, optimized routing, and built-in API capabilities perfect for healthcare platforms requiring fast, secure page loads and SEO optimization.

3.2 React Components - component-based UI library enabling reusable, maintainable interface elements and efficient state management - ideal for creating consistent healthcare dashboards and interactive patient portals.

3.3 Firebase Database - cloud-hosted NoSQL database providing real-time data synchronization, robust authentication, and HIPAA-compliant storage - essential for managing patient records and secure medical data.

1. **HealthVisualizer\_next Directory Structure**

HealthVisualizer\_next/

├── .next/

├── node\_modules/

├── public/

├── src/

4.1 .next: A build directory, managed by next.js, do not edit this folder.

node\_modules: External dependencies managed by npm or yarn, do not edit this folder.

public: Static assets for your app that are served directly without processing.

src: The core source code for the application, this will contain the pages, components etc.

1. **Source Directory Structure**

src/app/

├── components/ # Reusable UI components

├── doctorComponent/ # Doctor-specific UI components

├── mainComponent/ # Core and shared UI components

└── patientComponent/ # Patient-specific UI components

├── functionality/ # Core application logic

└── extracted/ # Directory for files extracted from external sources

├── pages/ # Application pages (routing)

├── appointments/ # Appointment management page (verify function)

├── DoctorScreen/ # Main screen for doctors

├── healthoverview/ # Overview of health information

├── LoginScreen/ # Login/registration screen

├── patientappointments/ # Appointments for a specific patient

├── patients/ # List of patients (for doctors?)

└── PatientScreen/ # Main screen for patients

├── globals.css # Global CSS styles

├── layout.js # App layout (header, footer, etc.)

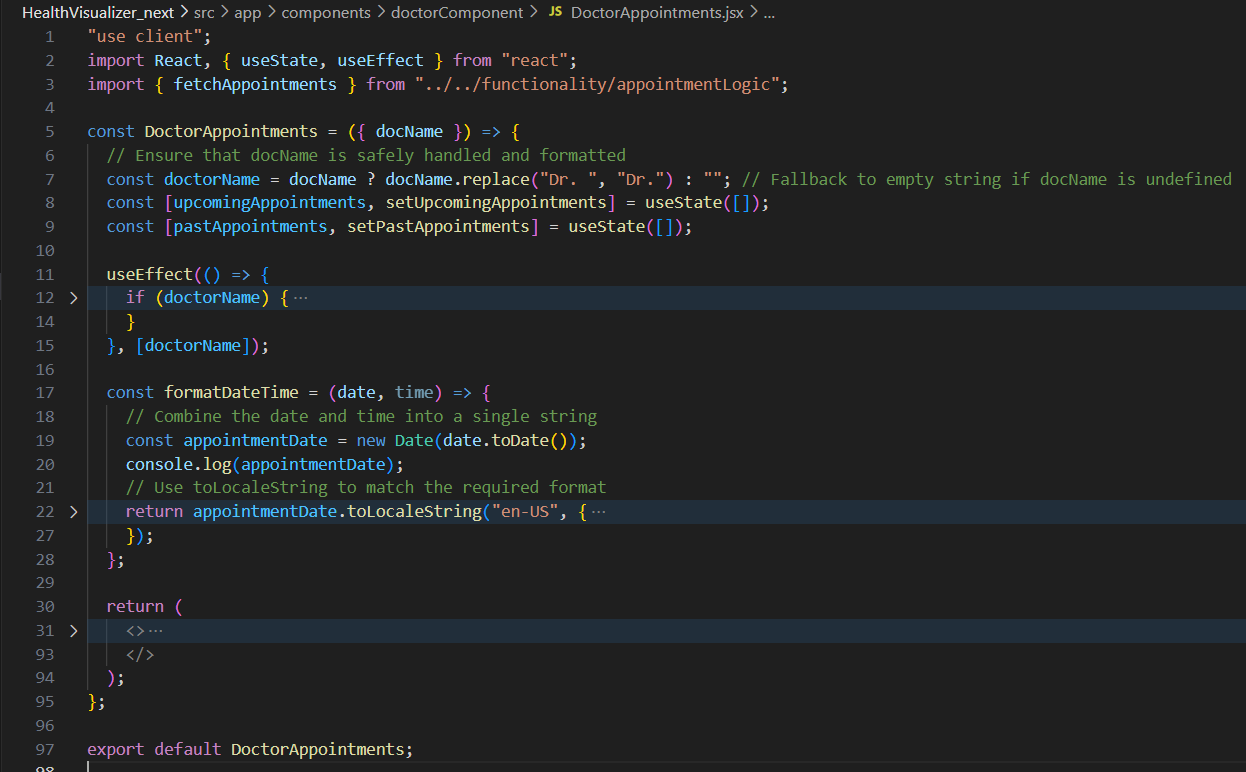
└── page.js # Application homepage

1. **Components Documentation:**

**DoctorAppointments.jsx:**

* Description: Manages and displays a doctor's upcoming/past appointments.
* Key Functionality: Fetches, categorizes, and formats appointment data. Renders the appointment lists (patient name, date/time, location, specialty). Handles "no appointments" message.
* Inputs (Props):
  + docName: string (required): Doctor's name (e.g., "Dr. test").
* State Variables:
  + upcomingAppointments: Array<Object>: Upcoming appointments.
  + pastAppointments: Array<Object>: Past appointments.
* useEffect Hook: Fetches appointments using fetchAppointments (from ../../functionality/appointmentLogic) when docName changes. Updates upcomingAppointments and pastAppointments.
* formatDateTime(date, time) function: Formats date/time (uses toLocaleString). Returns formatted string.
* Return Value (JSX): Renders "Upcoming Appointments" and "Past Appointments" sections, lists appointments, or displays "no appointments" message.

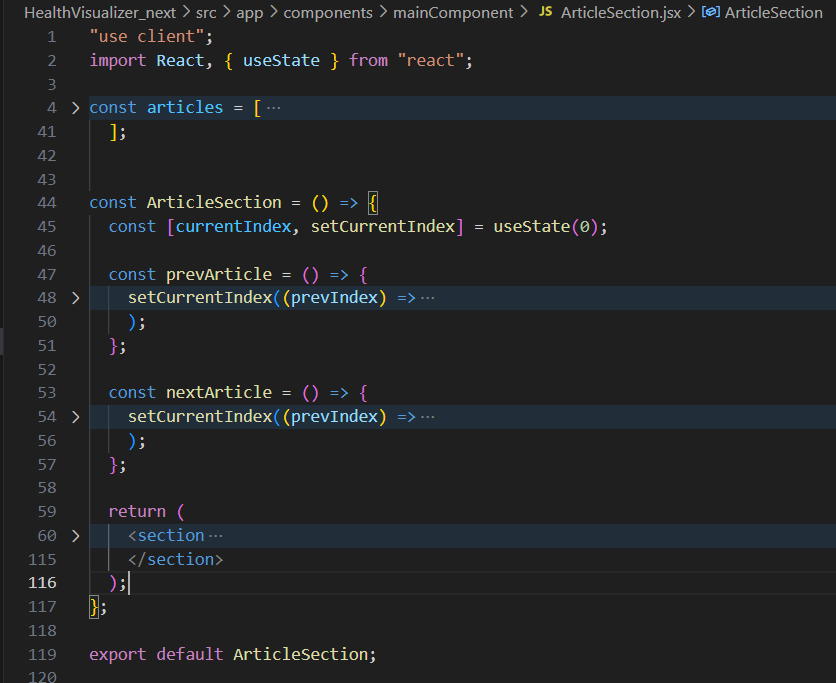
**Code:**



**ArticleSection.jsx:**

* Description: Displays a rotating section of health articles with titles, content, and images.
* Key Functionality:
  + Manages a list of articles with title, content, and image.
  + Rotates through the articles using "previous" and "next" buttons.
  + Renders the currently selected article's title, content, and image.
* Inputs (Props): None
* State Variables:
  + currentIndex: number: The index of the currently displayed article in the articles array. Initialized to 0.
* useEffect Hook: None
* prevArticle() function:
  + Description: Decrements the currentIndex to display the previous article. If the currentIndex is already at the beginning of the array, it loops to the end.
  + Parameters: None
  + Returns: None (updates the component's state)
* nextArticle() function:
  + Description: Increments the currentIndex to display the next article. If the currentIndex is already at the end of the array, it loops to the beginning.
  + Parameters: None
  + Returns: None (updates the component's state)
* Return Value (JSX):
  + Renders an <section> with a heading "Health Articles", a description, and a div containing the currently displayed article's content and image.
  + Includes "previous" and "next" buttons to navigate through the articles.

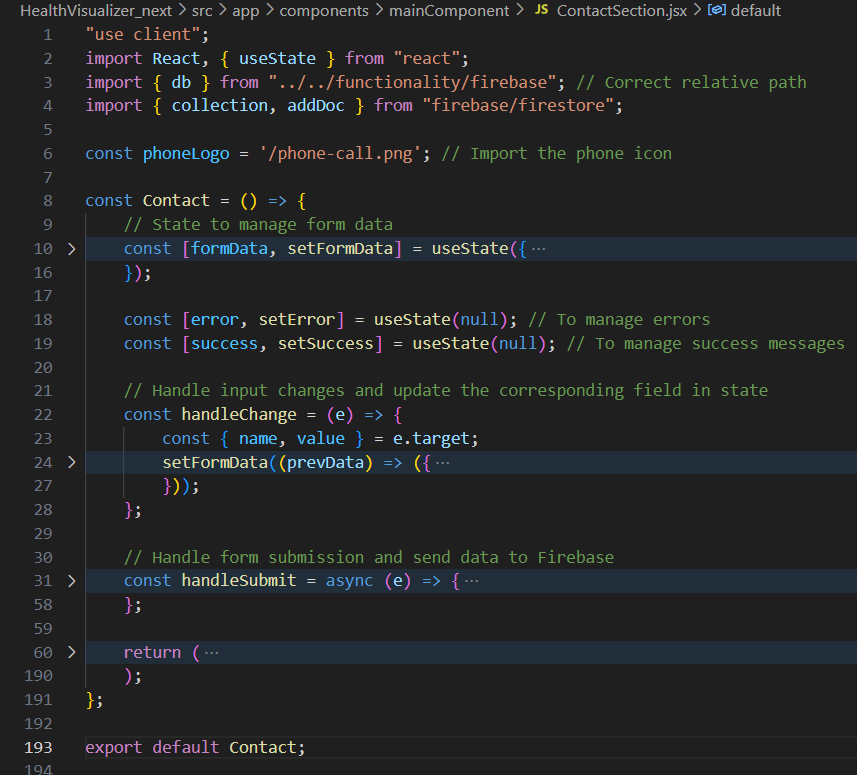
**Code:**

****

**ContactSection.jsx:**

* Description: Displays a contact form and contact information, allowing users to send messages to different departments.
* Key Functionality: Renders a contact form, manages form data, validates input, sends data to Firebase Firestore, and displays success/error messages.
* Inputs (Props): None
* State Variables:
  + formData: object: Stores the contact form data (firstName, lastName, email, department, message).
  + error: string: Stores any error message.
  + success: string: Stores any success message.
* useEffect Hook: None
* handleChange(e) function: Updates the formData state when input fields change.
* handleSubmit(e) function: Handles form submission, validates input, sends data to Firebase, and displays success/error messages.
* Return Value (JSX): Renders the contact form, contact information, phone number, and success/error messages.

**Code:**

****

**DarkMode.jsx:**

* Description: A component that allows users to toggle between light and dark mode.
* Key Functionality: Manages the dark mode state, toggles the dark mode class on the body element, and stores the preference in localStorage.
* Inputs (Props): None
* State Variables:
  + darkMode: boolean: Stores the current dark mode state (true for dark mode, false for light mode).
* useEffect Hook:
  + Initializes dark mode state based on localStorage and sessionStorage on component mount. Prevents resetting the mode on navigation.
* toggleDarkMode() function: Toggles the dark mode state, updates the darkMode state, adds/removes the dark class from the body element, and saves the preference in localStorage.
* Return Value (JSX): Renders a clickable div that acts as a toggle switch for dark mode.

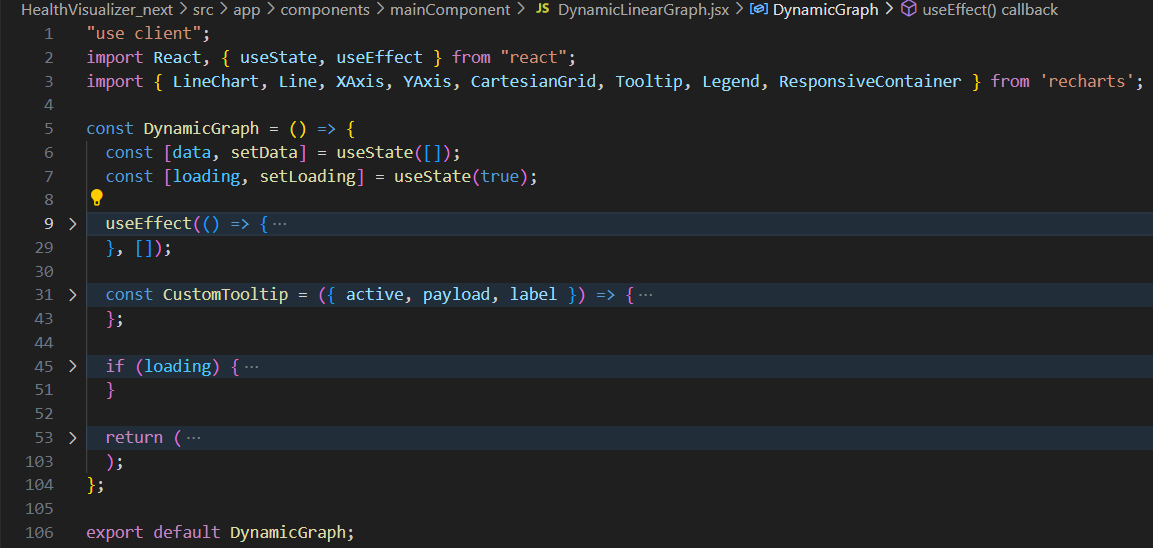
**Code:**

****

**DynamicLinearGraph.jsx:**

* Description: Fetches flu cases data from a JSON file and displays it as a dynamic line chart.
* Key Functionality:
  + Fetches data from /flu\_data\_2023\_2025.json.
  + Formats the data for use in a recharts LineChart.
  + Displays a loading message while fetching data.
  + Renders a responsive line chart using the recharts library, with custom tooltip.
* Inputs (Props): None
* State Variables:
  + data: Array<Object>: Stores the formatted data for the chart.
  + loading: boolean: Indicates whether the data is currently being fetched.
* useEffect Hook: Fetches data from /flu\_data\_2023\_2025.json on component mount. Formats the data and sets the data and loading state.
* CustomTooltip({ active, payload, label }) function: Custom tooltip for the recharts LineChart, displaying the date and number of cases when a data point is hovered over.
* Return Value (JSX): Renders a loading message while data is being fetched. Once data is loaded, it renders a responsive recharts LineChart displaying the flu cases data.

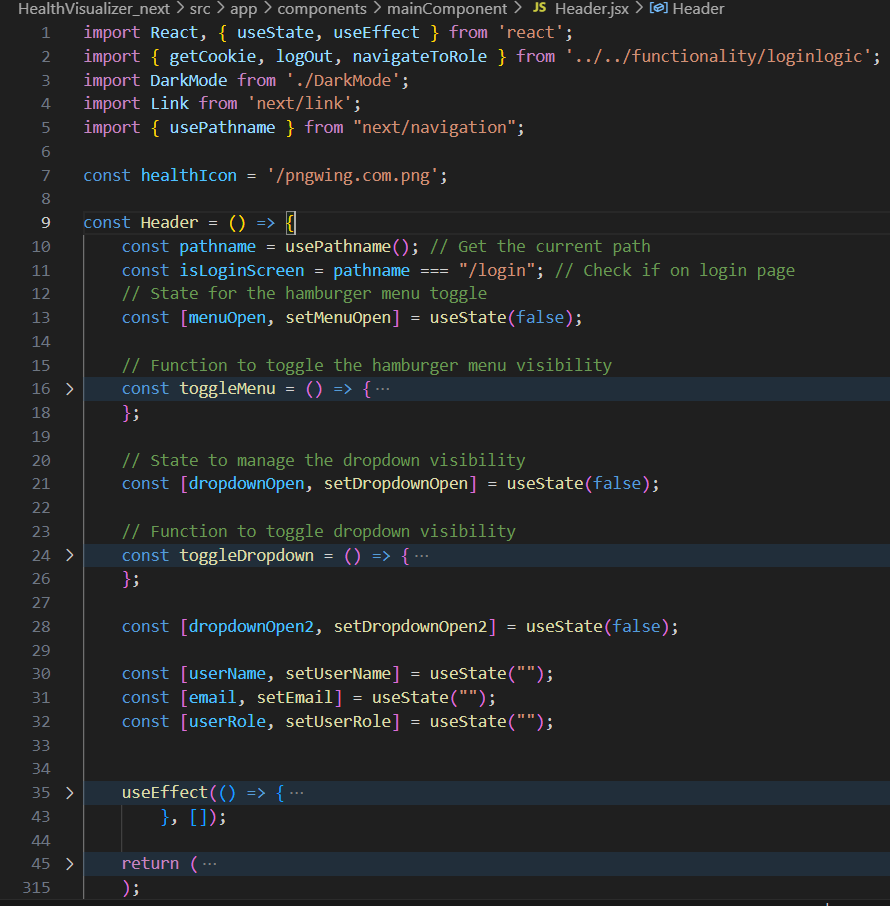
**Code:**

****

**Header.jsx:**

* Description: Renders the header/navigation bar for the application.
* Key Functionality:
  + Displays the logo and application title.
  + Provides navigation links to "Info", "Contact", "Health" (dropdown for "Statistics" and "Articles").
  + Handles user authentication: displays "Login" link if not authenticated, or username with logout/profile options in a dropdown if authenticated.
  + Toggles the display of navigation links on small screens (hamburger menu).
  + Includes the DarkMode component for toggling between light and dark modes.
* Inputs (Props): None
* State Variables:
  + menuOpen: boolean: Controls the visibility of the hamburger menu on small screens.
  + dropdownOpen: boolean: Controls the visibility of the "Health" dropdown menu.
  + dropdownOpen2: boolean: Controls the visibility of the "User" dropdown menu.
  + userName: string: Stores the username retrieved from cookies.
  + email: string: Stores the email retrieved from cookies.
  + userRole: string: Stores the user role retrieved from cookies.
* useEffect Hook: Fetches user data from cookies to display authenticated user information.
* toggleMenu() function: Toggles the state to set the variable menuOpen.
* toggleDropdown() function: Toggles the state to set the variable dropdownOpen.
* Return Value (JSX): Returns the application header, including the logo, navigation links, dark mode toggle, and conditional rendering for authentication and mobile menu.

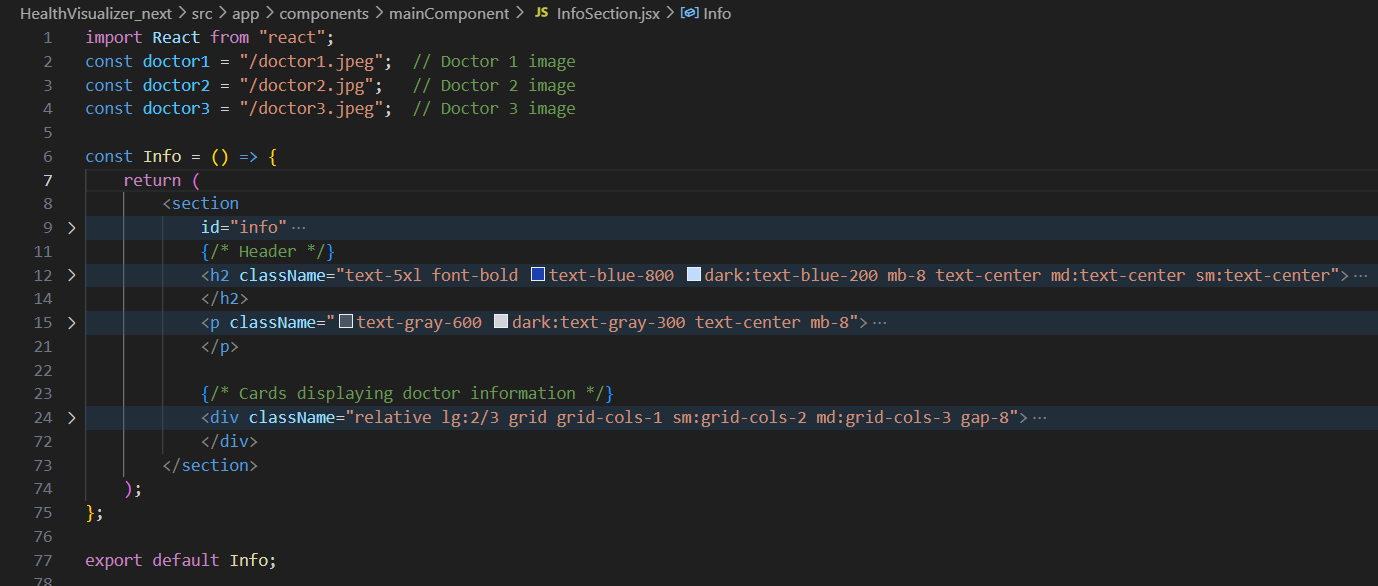
**Code:**



**InfoSection.jsx:**

* Description: Displays information about the team of doctors, including their names, specialties, and contact information.
* Key Functionality:
  + Displays a heading and introductory text.
  + Renders cards for each doctor, including their image, name, specialty, contact information, and email.
  + Uses a hover effect to display doctor information on the card.
* Inputs (Props): None
* State Variables: None
* useEffect Hook: None
* Return Value (JSX): Returns a section with information about the team members and their roles.

**Code:**

****

**LoginForm.jsx:**

* Description: Renders a basic login form with fields for email and password.
* Key Functionality: Displays input fields for email and password, taking controlled input.
* Inputs (Props):
  + email: string: The email input value.
  + setEmail: function: Function to update the email state.
  + password: string: The password input value.
  + setPassword: function: Function to update the password state.
* State Variables: None
* useEffect Hook: None
* Return Value (JSX): Returns a login form with email and password input fields.

**Code:**

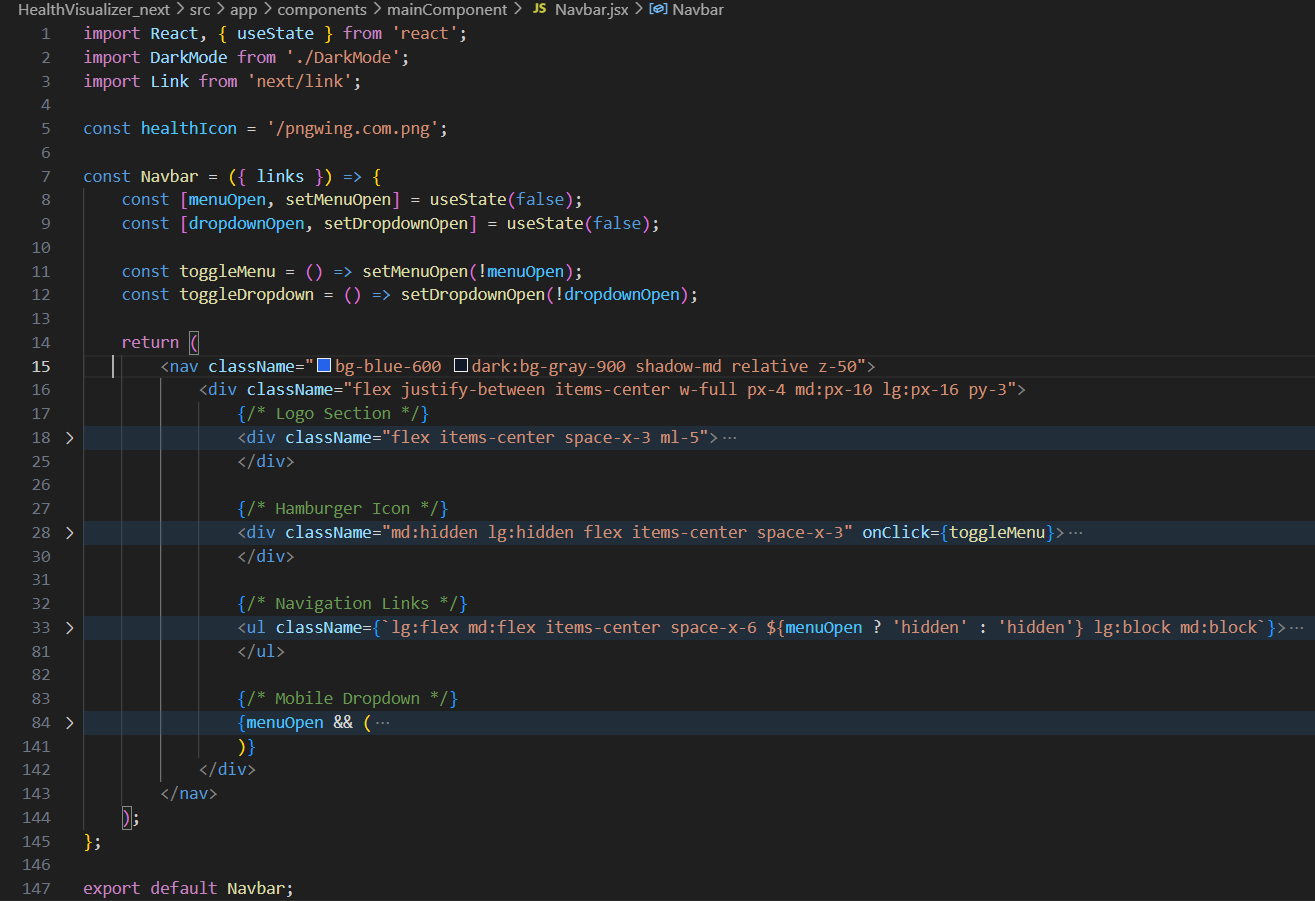
תמונה שמכילה טקסט, צילום מסך, תוכנה

התיאור נוצר באופן אוטומטי

**Navbar.jsx:**

* Description: Renders the navigation bar for the application.
* Key Functionality:
  + Displays the logo and application title.
  + Provides navigation links based on the provided links prop.
  + Handles hamburger menu functionality for small screens.
* Inputs (Props):
  + links: array: Array of link objects, each with a name and href (for regular links) or name and dropdownItems (for dropdowns). Dropdown items contain name and onClick functions.
* State Variables:
  + menuOpen: boolean: Controls the visibility of the hamburger menu on small screens.
  + dropdownOpen: boolean: Controls the visibility of the dropdown menu.
* useEffect Hook: None
* toggleMenu() function: Toggles the hamburger menu's visibility state.
* toggleDropdown() function: Toggles the dropdown menu's visibility state.
* Return Value (JSX): Returns the application navigation bar, including the logo, navigation links, mobile menu, and dark mode toggle.

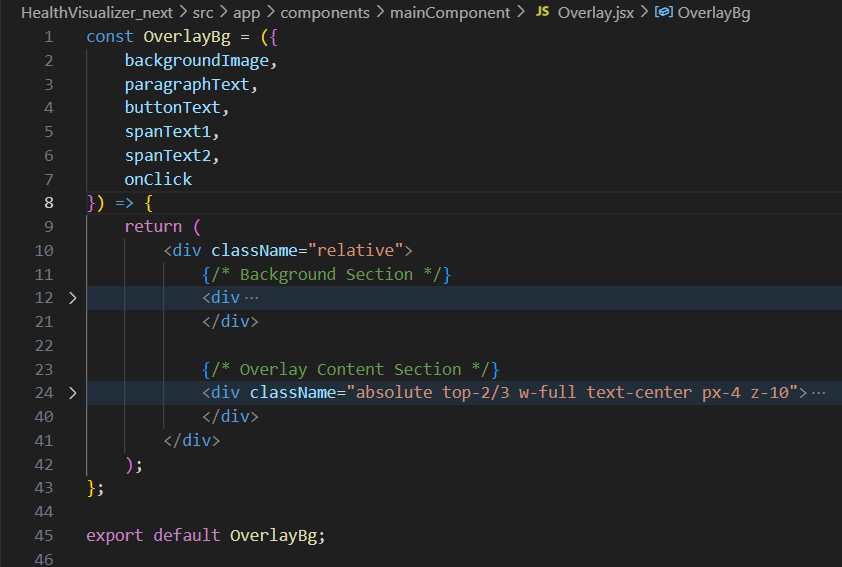
**Code:**

****

**OverlayBg.jsx:**

* Description: Renders an overlay with a background image, text, and a button.
* Key Functionality: Displays a section with an image, title, paragraph, and a button.
* Inputs (Props):
  + backgroundImage: string: URL of the background image.
  + paragraphText: string: Text for the paragraph.
  + buttonText: string: Text for the button.
  + spanText1: string: The first part of the title.
  + spanText2: string: The second part of the title.
  + onClick: function: Function to call when the button is clicked.
* State Variables: None
* useEffect Hook: None
* Return Value (JSX): Returns a section with a background image, title, paragraph, and a button.

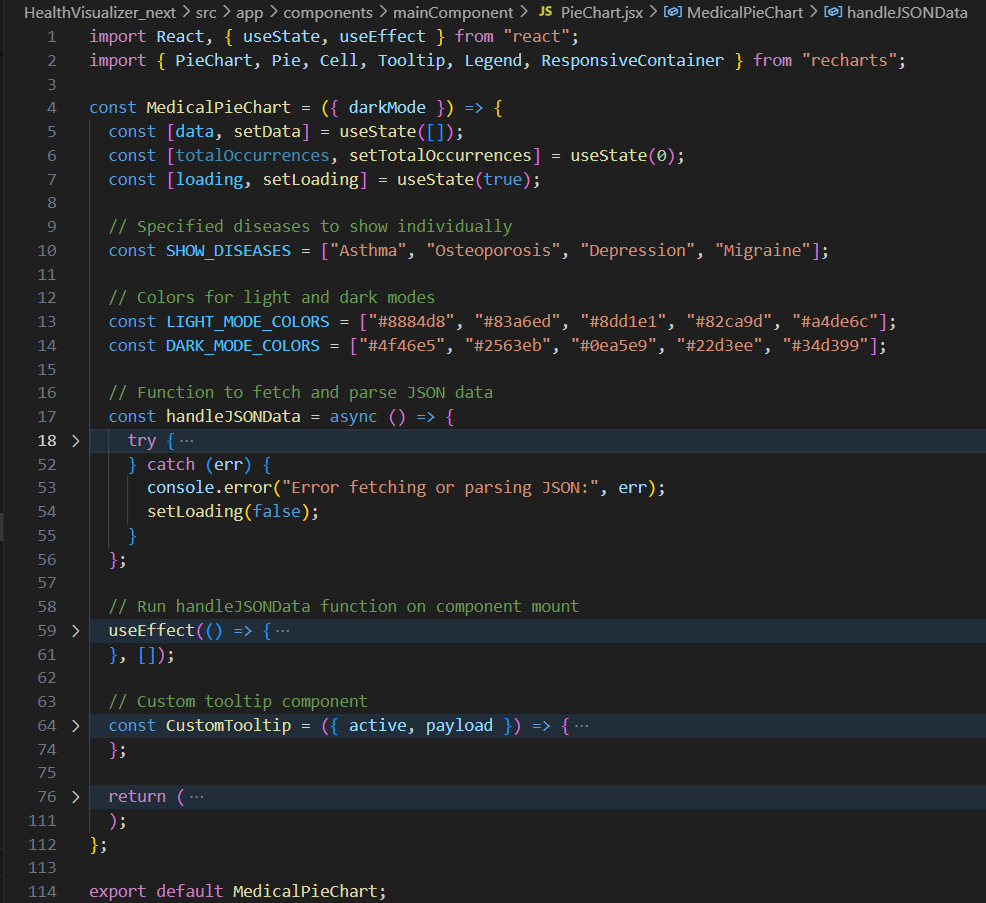
**Code:**

****

**MedicalPieChart.jsx:**

* Description: Displays a pie chart showing the distribution of selected medical conditions.
* Key Functionality: Fetches data from a JSON file, processes the data, and renders a pie chart using Recharts.
* Inputs (Props):
  + darkMode: boolean: whether dark mode is on
* State Variables:
  + data: array: Array of objects containing the data for the pie chart.
  + totalOccurrences: number: The total number of occurrences of the medical conditions (not really used).
  + loading: boolean: Indicates whether the data is currently being fetched.
* useEffect Hook: Fetches data from medical\_profiles.json on component mount and formats the data for the pie chart.
* CustomTooltip({ active, payload }) function: Custom tooltip for the Recharts PieChart.
* Return Value (JSX): Renders a title, loading message, and the pie chart component itself using recharts.

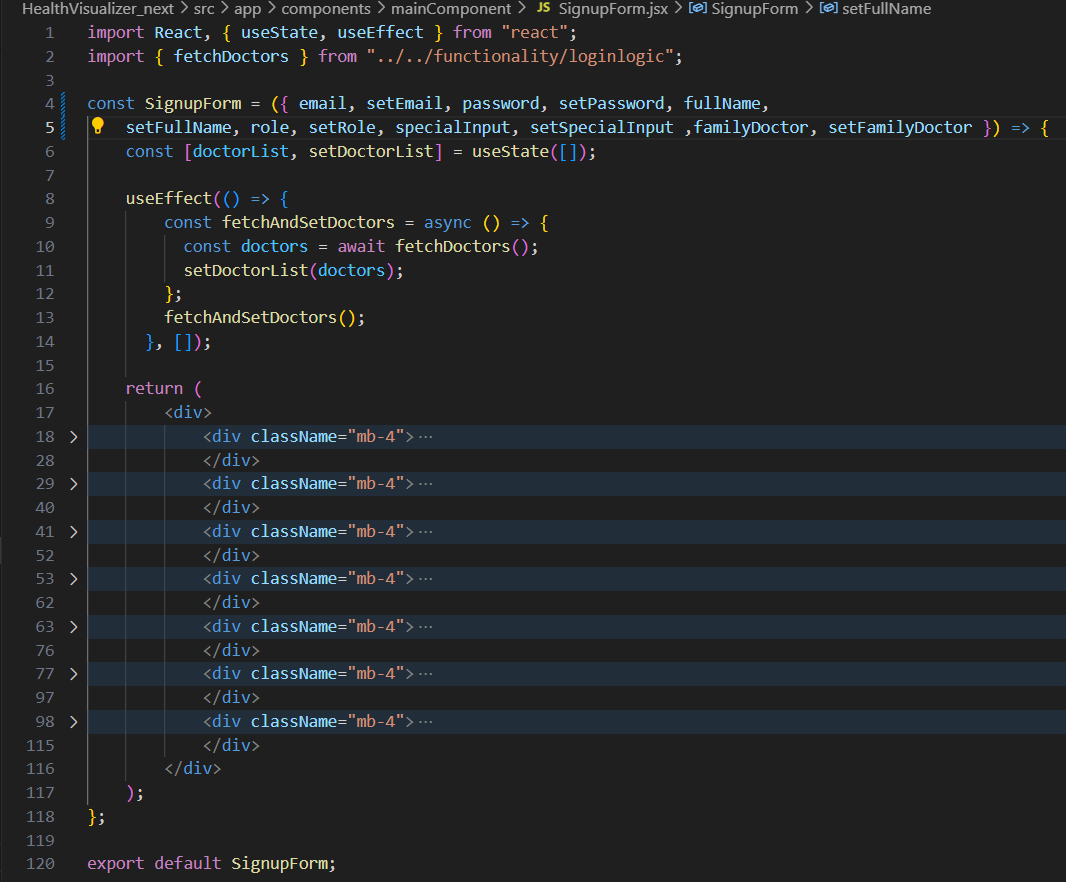
**Code:**

****

**SignupForm.jsx:**

* Description: Renders a signup form with fields for name, email, password, role, and a special input based on the selected role.
* Key Functionality: Displays form input fields, including conditional rendering based on the selected user role.
* Inputs (Props):
  + email: string: The email input value.
  + setEmail: function: Function to update the email state.
  + password: string: The password input value.
  + setPassword: function: Function to update the password state.
    - fullName: string: The name input value.
  + setFullName: function: Function to update the name state.
  + role: string: The password input value.
  + setRole: function: Function to update the password state.
* State Variables:
  + None
* useEffect Hook: None
* Return Value (JSX): Returns a registration form.

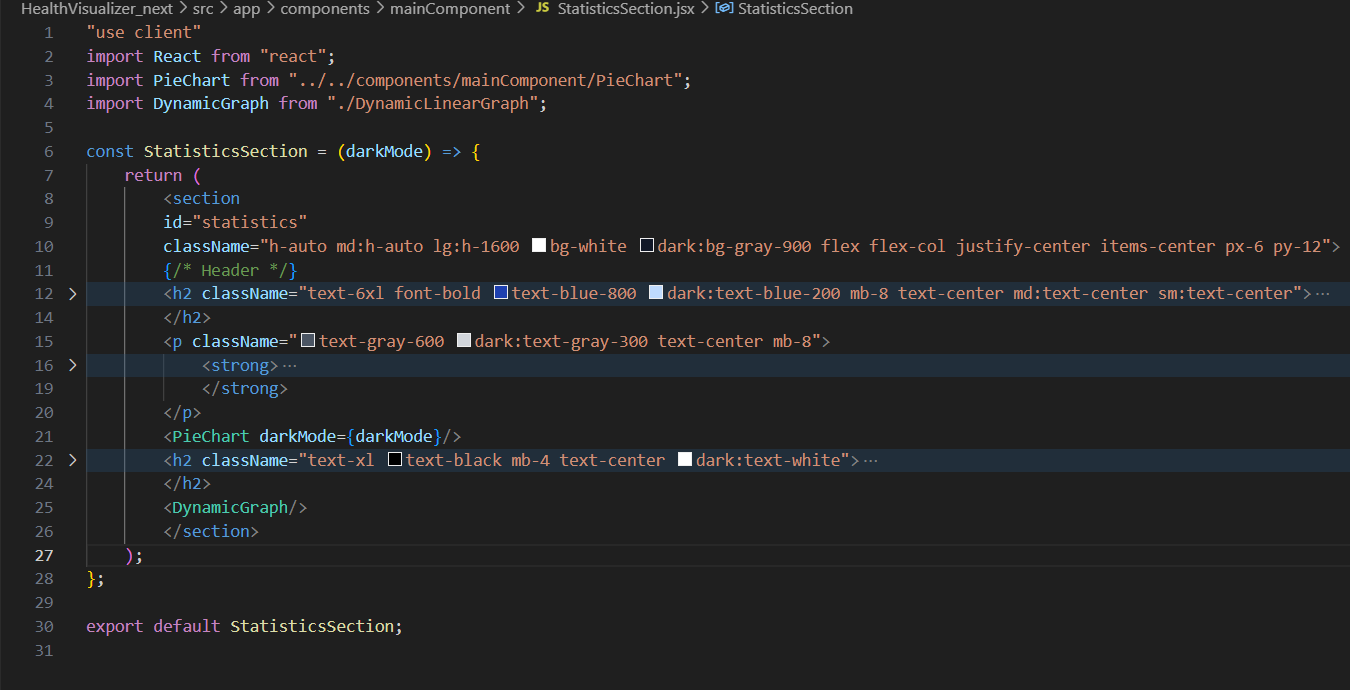
**Code:**

****

**StatisticsSection.jsx:**

* Description: Renders a section displaying a heading and a dynamic graph.
* Key Functionality: Displays the graph about statistics and renders the entire statistics section.
* Inputs (Props):
  + darkMode: boolean: whether dark mode is on
* State Variables:
  + None
* useEffect Hook: None
* Return Value (JSX): Returns a title and description of the section.

**Code:**

****

**Patient\_Appointments.jsx:**

* Description: Displays a list of upcoming appointments for a patient and allows them to add new appointments.
* Key Functionality:
  + Fetches a list of appointments from Firebase Firestore.
  + Filters appointments based on a search query and date.
  + Allows users to add new appointments through a pop-up form.
* Inputs (Props): None
* State Variables:
  + doctors: array: List of available doctors.
  + appointments: array: List of user's upcoming appointments.
  + newAppointment: object: Details of the new appointments (e.g. doctorName, Date).
  + showAddAppointment: boolena: To show or not to show the add appointment component.
  + loading: boolean: for showing loader component.
  + error: string: string message of error.
  + searchQuery: search box value to search for results
    - filterDate: filter the values from this date.
* useEffect Hook:  
  \*Fetches the doctor array if it has a doctor component and also fetch all the appointments.
* handleCreateAppointment(e) function: Creates an appointment if certain parameters are met. Also create it in the firebase
* Returns a list of parameters to the upcoming appointments component.

**BMI\_Claculator.jsx:**

* Description: Allows users to calculate their BMI (Body Mass Index) and provides information about their BMI category.
* Key Functionality:
  + Takes the height and weight of the user as input.
  + Calculates the BMI based on the height and weight values.
  + Determines the BMI category (Underweight, Normal weight).
* Inputs (Props): None
* State Variables:
  + height: string: User's height in centimeters.
  + weight: string: User's weight in kilograms.
  + bmi: float: Calculated Body Mass Index
    - category: The category the BMI user is in.
    - explanation: Explain the situation of the user to their current situation in their respective categories.
    - setAdvis: what are good instructions to follow for this kind of user.
      * savedBmi: to save the value that was just stored in this time.
      * userName: The name of the user from the cookies or guess it is not here.
  + userEmail: the user's e mail from the cookies.
* useEffect Hook: retrieves user e mail and user names from the cookies.
* BMIcalc() function: does the caluclation of the BMI for the user.
* saveBmi(): for saving the user information to the cloud.
* Return Value (JSX): Returns BMI form.

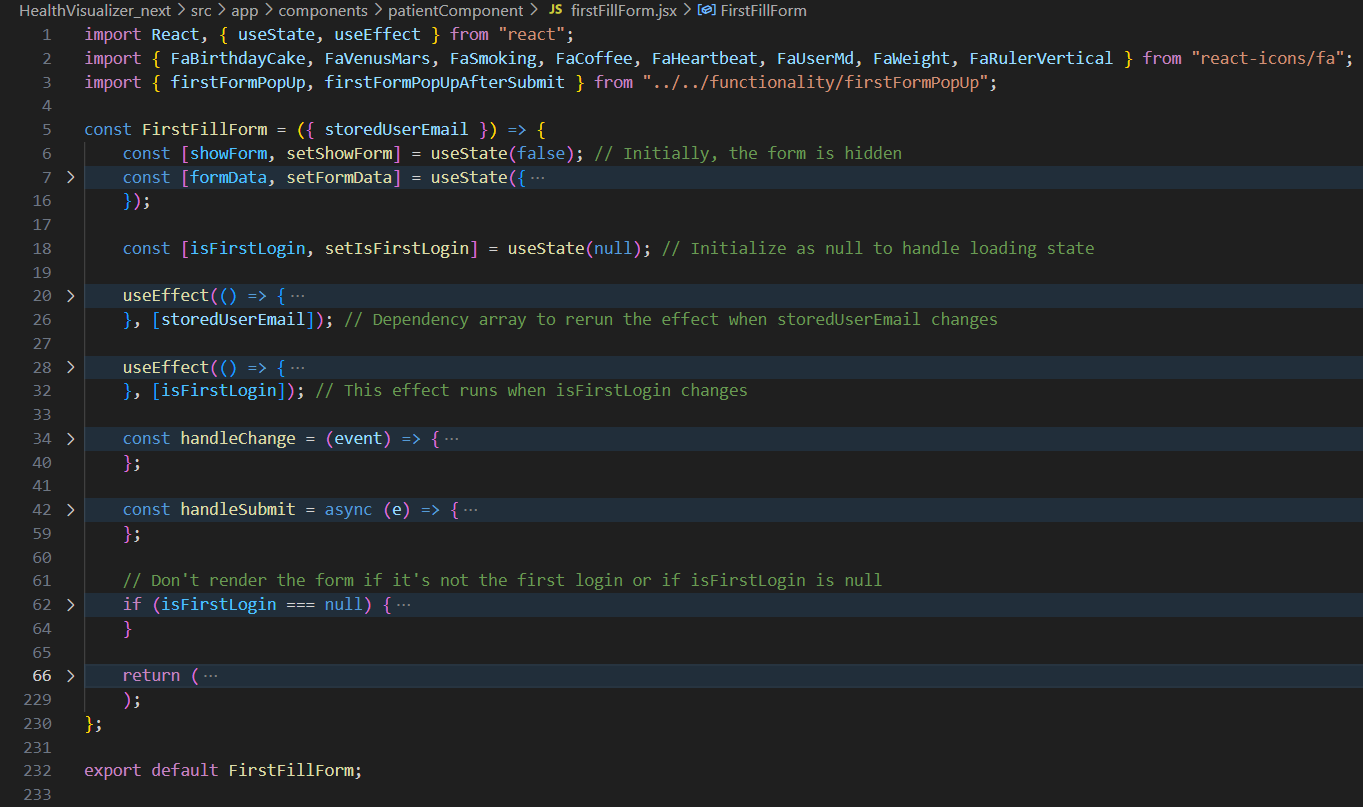
**Code:**



**FirstFillForm.jsx:**

* Description: Collects initial health information from new users.
* Key Functionality: Displays a pop-up form for new users to enter their health information (age, gender, height, weight, smoking habits, coffee consumption, and general health status).
* Inputs (Props):
  + storedUserEmail: string: The email of the user who is filling the form.
* State Variables:
  + showForm: boolean: Controls whether the form is displayed or not.
    - formData: An object to hold the various form fields.
    - isFirstLogin: check if the user has ever logged in or not
* useEffect Hook:  
  \*Used to get the new first login flag from the cloud  
  \*handleChange() : updates the form data on what was filled.  
  \*handleSubmit(): submit the form and also validates that we got everything corectly.
* Return Value (JSX): Returns returns form which include health and other information

**Code:**

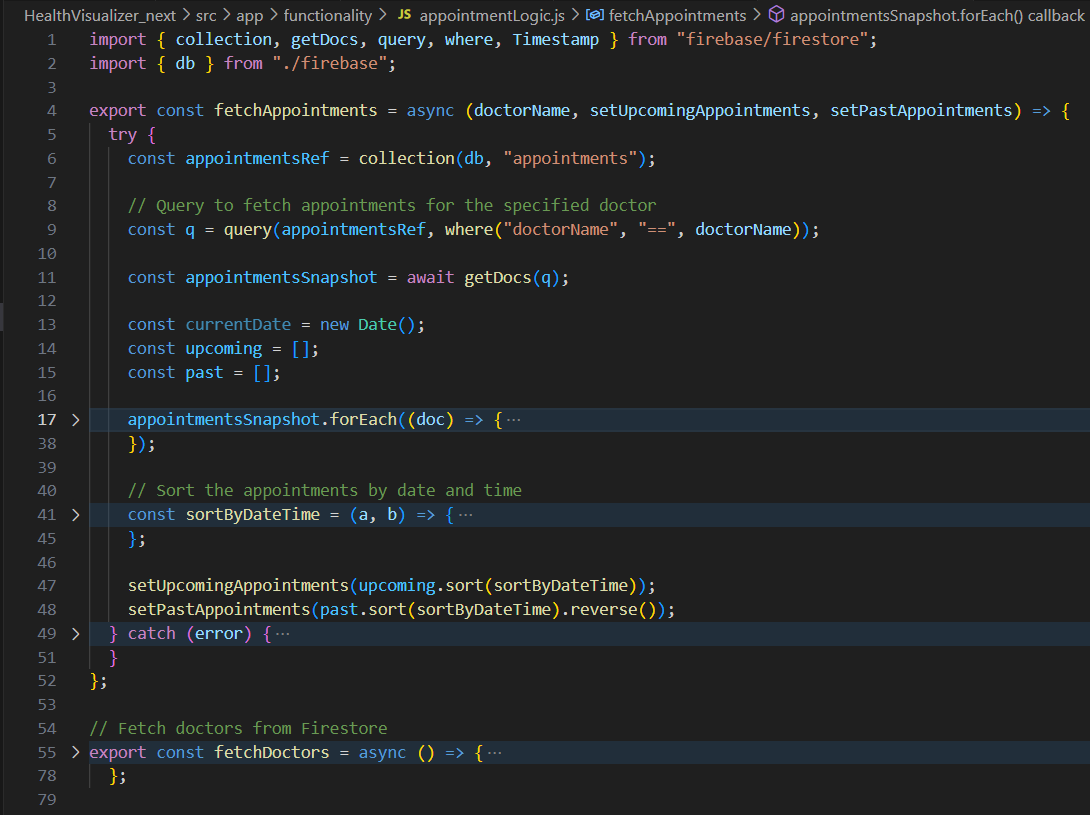
****

1. **Functionality Documentation:**

**appointmentLogic.js:**

* Description: Contains functions for managing and fetching appointment data from Firebase Firestore.
* Key Functionality:
  + fetchAppointments(doctorName, setUpcomingAppointments, setPastAppointments): Fetches upcoming and past appointments for a specified doctor from Firestore.
  + fetchDoctors(): Fetches a list of doctors from Firestore.
* Inputs (Props): Nones
* fetchAppointments Function:
  + Parameters:
    - doctorName: string: The name of the doctor to fetch appointments for.
    - setUpcomingAppointments: function: A function to update the state with the upcoming appointments.
    - setPastAppointments: function: A function to update the state with the past appointments.
* fetchDoctors Function: Does not get inputs. It just return list of doctors.
* Return Value:  
  \*fetchAppointments: The returned function updates states of past and the upcoming appointments.
* \*fetchDoctors: List of doctors.

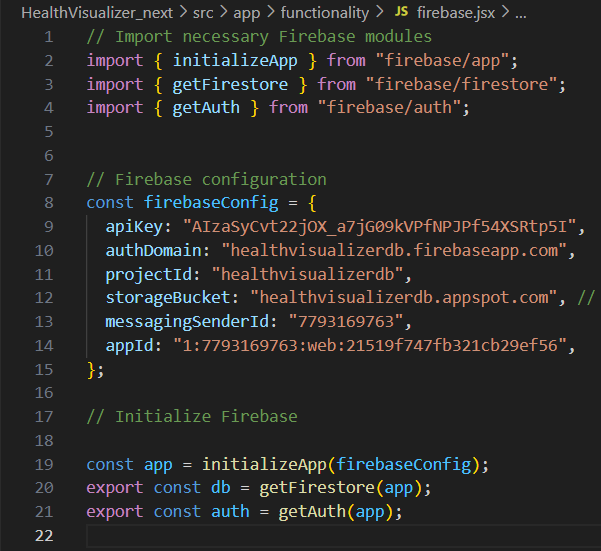
**Code:**



**firebase.js:**

* Description: Initializes the Firebase application and exports instances of Firestore and Authentication.
* Key Functionality:
  + Initializes the Firebase app with the provided configuration.
  + Exports instances of Firestore (db) and Authentication (auth) for use in other parts of the application.
* Inputs (Props): None
* State Variables: None
* useEffect Hook: None
* Return Value:
  + Exports db: Instance of Firestore
  + Exports auth: Instance of Firebase Authentication

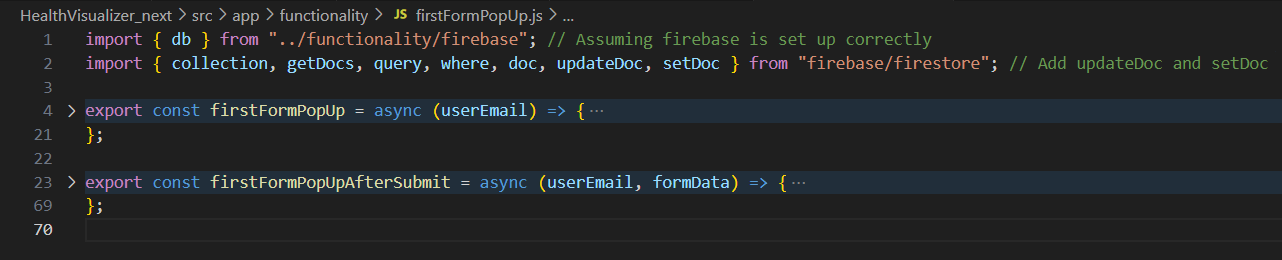
**Code:**

****

**firstFormPopUp.js:**

* Description: Contains functions to manage the first login form for new users.
* Key Functionality:
  + firstFormPopUp(userEmail): Checks if a user is logging in for the first time by querying their record in the "users" collection and checking the "firstLogin" field.
  + firstFormPopUpAfterSubmit(userEmail, formData): Handles the submission of the first login form, updating the "firstLogin" field to false and creating a new document in the "patients" collection with the form data.
* Inputs (Props): None
* firstFormPopUp(userEmail) Function:
  + Parameters:
    - userEmail: string: The email of the user.
  + Return Value:
    - Returns the value of the firstLogin field from the user document.
* firstFormPopUpAfterSubmit(userEmail, formData) Function:
  + Parameters:
    - userEmail: string: The email of the user submitting the form.
    - formData: object: Object containing the data submitted in the first login form (age, gender, height, weight, etc.).
  + Return Value:
    - Returns true if the form submission and processing were successful, and false if an error occurred or no user was found with the provided email.

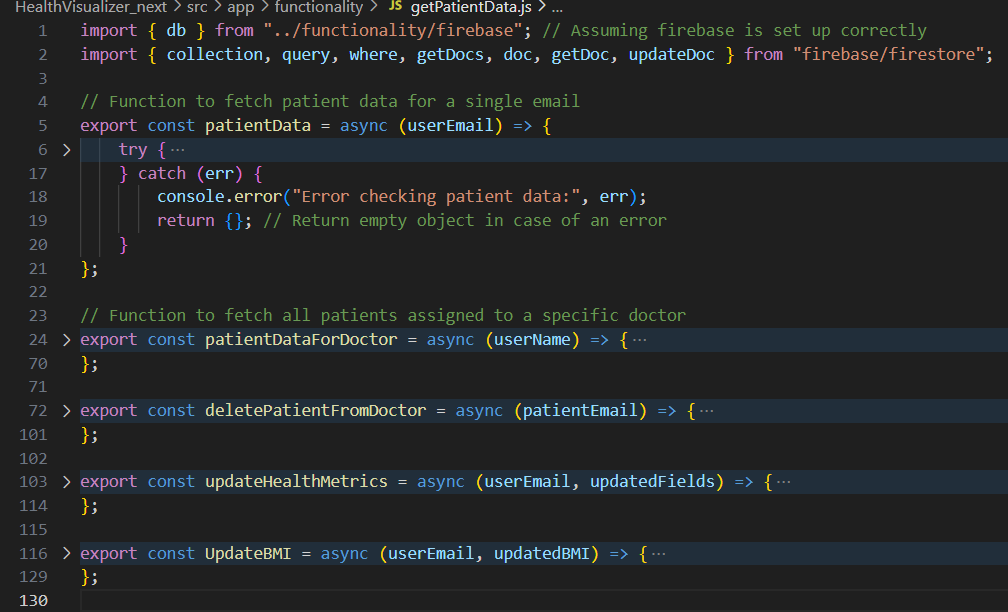
**Code:**

****

**getPatientData.js:**

* Description: Contains functions for fetching and managing patient data from Firebase Firestore.
* Key Functionality:
  + patientData(userEmail): Fetches patient data for a single patient, given their email.
  + patientDataForDoctor(userName): Fetches patient data for all patients assigned to a specific doctor, given the doctor's username.
  + deletePatientFromDoctor(patientEmail): It unassigns the patient and no longer connect the family doctor field
  + updateHealthMetrics(userEmail, updatedFields): Updates specific health metrics in a patient's document, given their email and an object containing the fields to update.
  + UpdateBMI(userEmail, updatedBMI): Updates Body Mass Index in the doctor
* Inputs (Props): None
* Functions:  
  \* deletePatientFromDoctor: (input): Patient email and returns promise (async)  
  \* patientDataForDoctor :(Input): family doctor name and returns patient's array or rejected promise  
  \* updateHealthMetrics (userEmail, updatedFields): (input)  
  \*User Email: for the users component  
  \*updatedFieds: the health metrics as properties and the value which need to be update to.
* patientData(userEmail) Function:
  + Parameters:
    - userEmail: string: The email of the patient.
  + Return Value:
    - Returns an object containing the patient's data, or an empty object if no data is found or an error occurs.
* patientDataForDoctor(userName) Function:
  + Parameters:
    - userName: string: The username of the doctor.
  + Return Value:
    - Returns an array of patient data objects, or an empty array if no patients are assigned to the doctor or an error occurs.

**Code:**

****

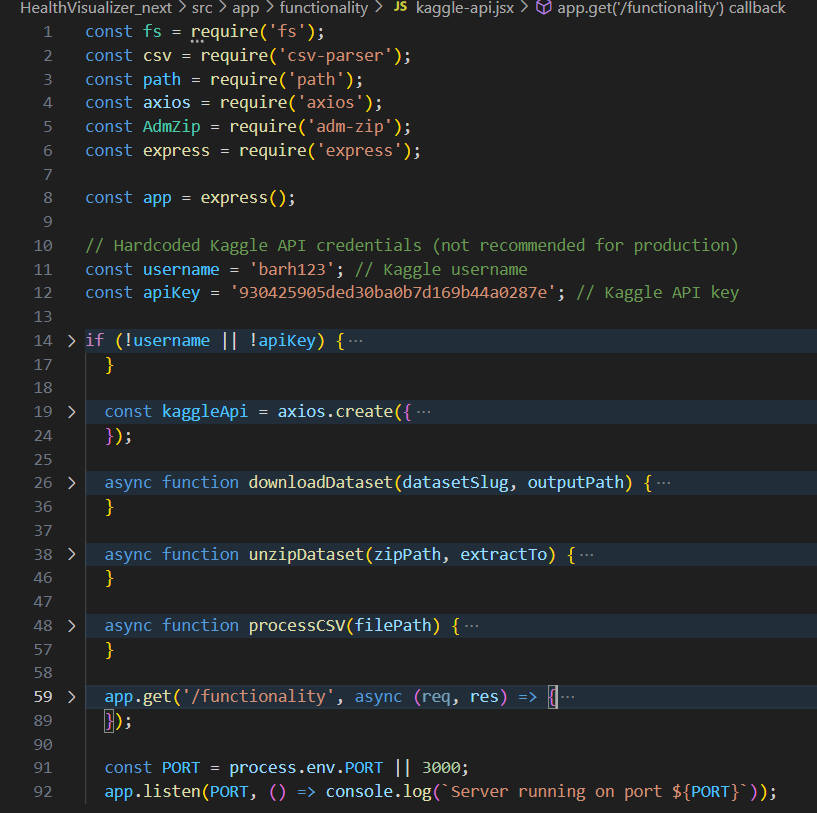
**kaggle-api.jsx:**

* Description: This script downloads, unzips, and processes a healthcare dataset from Kaggle, calculates the number of asthma patients and sends some other parameters such as: number of asthma people, totla patients
* Key Functionality:

1. Downloads data from a source
2. Unzipp all of the data
3. Process all of the CSV data
   * The script requires Kaggle API credentials (username and API key) to download datasets.
   * It uses Axios for making HTTP requests to the Kaggle API.
   * AdmZip is used for unzipping the downloaded dataset.
   * The script reads a CSV file from the unzipped directory.

* Inputs (Props): None
* Functions:  
  \*This js has more functions which I'll mention here:  
  downloadDataset(datasetSlug, outputPath);  
  unzipDataset(zipPath, extractTo);  
  processCSV(filePath);
* Return Value:  
  Returns json result of a file which includes: asthmaCount, totalCount

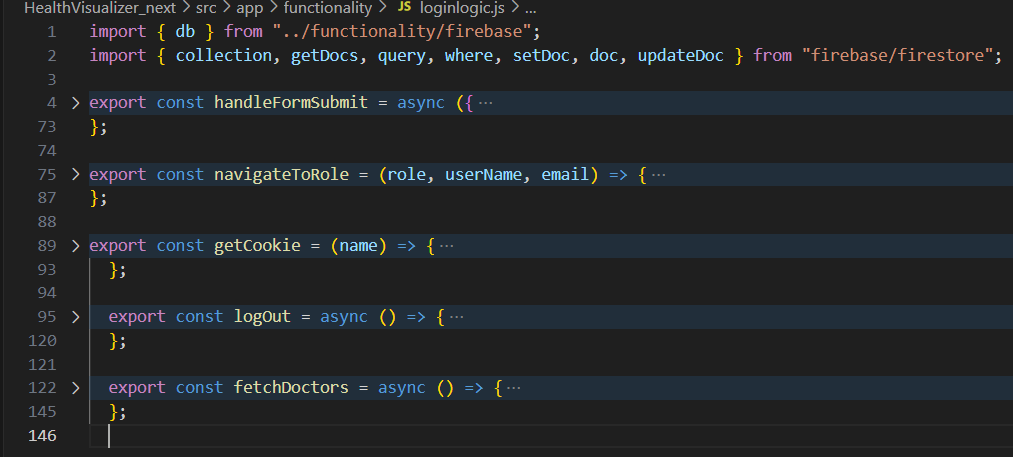
**Code:**

****

**loginlogic.js:**

* Description: Contains functions for handling user login, signup, navigation, cookie management, and user data fetching.
* Key Functionality:
  + handleFormSubmit(...): Handles both user login and signup logic, interacting with Firebase Firestore for authentication and data storage.
  + navigateToRole(role, userName, email): Navigates the user to the appropriate screen based on their role (Doctor or Patient) and sets cookies for user information.
  + getCookie(name): Retrieves the value of a specific cookie.
  + logOut(): Logs out the user by updating their status in Firestore and clearing the user's cookies.
  + fetchDoctors(): Fetches a list of doctors from Firestore.
* Inputs (Props): None
* Functions:  
  \* handleFormSubmit()(input): parameters including user data and returns a promise.  
  \* navigateToRole()(Input) :user name role and email (returned from getCookie)  
  \* getCookie(name)(Input):cookie name (returns parameter to role)
* Return Value:  
  The returned variables include setting of the users function.

**Code:**

****

1. **Pages Documentation:**

pages/appointments/page.jsx:

* Description: Displays a list of upcoming appointments for a doctor and allows them to manage the appts.
* Key Functionality: The page presents the ability to see the upcoming appts
* Inputs (Props): None
* State Variables:  
  \*userName: The name of the doctor , that is the patient  
  \*email: String  
  \*showAddAppointment: to show or not to show  
  \* appointments:  
  \* doctor:
  + useEffect Hook: It retrieves all the needed infromation  
    \*The links array gives name and onClick
* Return Value: HTML component

pages/DoctorScreen/page.jsx:

* Description: This page is for the doctor , it presents patient's infromation
* Key Functionality: Contains navigation , to patients
* Inputs (Props):
* \* userName : the name of the doctor, that is the doctor
* \* email:email for the user
* \* searchTeam: to search with
* \* appointment: is for the component to get.
* \* notification: for notifications
  + useEffect Hook: retrieve users for the doctor.  
    \*The links array gives name and onClick
* Return Value: Returns html component

pages/healthoverview/page.jsx:

* Description: It's a component to display information about the patients.
* Key Functionality: Manages the current user's health information.
* Inputs (Props): None
* State Variables:  
  \*userName from the cookies.  
  \* userEmail: string | null - for all of the information below  
  \* allPatientData: The general and all user data from the component
* useEffect Hook: Retrieves user information from the cookies and set it here.
* The links array gives name and onClick
* Return Value: returns the HTLM component

pages/LoginScreen/page.jsx:

* Description: Creates the whole login and account register for the project
* Key Functionality: Create a state hook to help us to toggle between the login and register.
* Inputs (Props): None
* The links array gives name and onClick
* Return Value: Creates the base for the registration page.

pages/patientappointments/page.jsx:

* Description: Create a list of Patient Appointments
* Key Functionality: Show data and use getCookies. Also show appointments.
* Inputs (Props): None
* State Variables:
  + userName
* useEffect Hook:  
  Sets user to local function
* Returns: Has all the html to create list

pages/patients/page.jsx:

* Description: Patients Page
* Key Functionality: Render html for patient data or no patients data
* Inputs (Props): Nones
* State Variables:  
  \*useEffectHook():
  + Returns The general format of pages

pages/page.js:

* Description: General HTML of the pages. The user sees something and can interact with it.
* Key Functionality: Setting HTML values like links or titles.
* Inputs (Props): Nones
* State Variables:  
  None
* useEffectHook: No one
* Return Value:HTML Component

globals.css:

* Description: Contains global CSS styles for the application.
* Key Functionality: Defines default styles for elements, typography, colors, and potentially global CSS variables.
* Inputs (Props): None
* Return Value: Does not have an

**9. API Documentation, Database Links:**

* **Firebase:**
  + Firebase SDK for JavaScript: [https://firebase.google.com/docs/web/setup](https://www.google.com/url?sa=E&q=https%3A%2F%2Ffirebase.google.com%2Fdocs%2Fweb%2Fsetup)
  + Firestore: [https://firebase.google.com/docs/firestore](https://www.google.com/url?sa=E&q=https%3A%2F%2Ffirebase.google.com%2Fdocs%2Ffirestore)
  + Firebase Authentication: [https://firebase.google.com/docs/auth](https://www.google.com/url?sa=E&q=https%3A%2F%2Ffirebase.google.com%2Fdocs%2Fauth)
  + Database Link: <https://console.firebase.google.com/project/healthvisualizerdb/firestore/databases/-default-/data/~2Fappointments~2F10bPXQaMFh6j5i5aHbXa>
* Kaggle API: [https://github.com/Kaggle/kaggle-api](https://www.google.com/url?sa=E&q=https%3A%2F%2Fgithub.com%2FKaggle%2Fkaggle-api)

**10.Key Third-Party Libraries:**

* **Next.js:** Library that implements all server-side calls.
* **React:** Main client-side JavaScript library.
* **Tailwind CSS:** Utilized for styling.