МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

Кафедра веб-технологий и компьютерного моделирования

-

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ПОСТАНОВКИ ДИАГНОЗА С КОМПОНЕНТОЙ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЯ

Курсовая работа

Царя Алексея Александровича
студента 2 курса,
специальности 1-31 03 08-01
«Математика и информационные технологии»

Научный руководитель:
доцент, кандидат технических наук
И.Р. Лукьянович

Минск, 2024

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Современный мир стал свидетелем значительного развития информационных технологий и интернет-сервисов, которые проникают во все сферы нашей жизни. Медицина не является исключением, и в настоящее время существует большое количество онлайн-ресурсов, предлагающих медицинскую информацию и консультации. Однако, при постановке диагноза пациенты часто сталкиваются с трудностями и неуверенностью в правильности своих выводов.

В этом контексте разработка веб-приложения, способного поддержать принятие решений при постановке диагноза, становится актуальной задачей. Такое приложение может стать незаменимым инструментом для пациентов, позволяющим им получить быстрый и точный анализ своих симптомов, а также персонализированные рекомендации по дальнейшим действиям.

Целью данной курсовой работы является разработка веб-приложения под названием "MedExpert", которое будет выступать в роли персонального врача в онлайн-формате. "MedExpert" будет предлагать пользователям возможность ввести свои симптомы и получить предварительный диагноз, основанный на анализе этих симптомов. Кроме того, приложение будет предоставлять рекомендации по дальнейшим действиям.

В ходе работы будет рассмотрена архитектура и основные компоненты приложения, а также применяемые методы анализа симптомов и генерации рекомендаций. Будет проведено исследование существующих подходов к поддержке принятия решений в медицине и выбраны наиболее подходящие для реализации в рамках "MedExpert".

Таким образом, данная курсовая работа имеет практическую значимость, так как предлагает разработку веб-приложения, способного помочь пользователям в постановке предварительного диагноза и принятии обоснованных решений по дальнейшему лечению и обращению за медицинской помощью. Результаты работы могут быть использованы для улучшения доступности и качества медицинского обслуживания в онлайн-среде.

В процессе выполнения курсовой работы будут рассмотрены основные концепции и принципы, связанные с React. Будут изучены способы создания компонентов на React, работа с состоянием и свойствами компонентов, а также использование React Router для управления навигацией.

Далее будет рассмотрено создание серверных приложений на NodeJS, использование фреймворка Express для создания API и обработки запросов от клиента. Будет изучено использование баз данных MySQL для хранения данных приложения.

В рамках данной курсовой работы будет проведен основательный анализ основных понятий и принципов, связанных с популярной библиотекой React. Будет уделено внимание изучению различных подходов к созданию компонентов с использованием React, а также методике работы с состоянием и свойствами компонентов. Особое внимание будет уделено изучению React Router, инструмента, позволяющего эффективно управлять навигацией в веб-приложениях.

Далее будет проведено исследование создания серверных приложений с использованием Node.js, платформы, позволяющей выполнять JavaScript-код на стороне сервера. Будет изучен фреймворк Express, который предоставляет удобные инструменты для создания API и обработки запросов от клиента. Будет уделено внимание основным принципам и методам работы с Express.

Кроме того, в рамках работы будет рассмотрено использование баз данных MySQL для хранения данных веб-приложения. Будет проведен анализ методов взаимодействия с базой данных, таких как создание таблиц, выполнение запросов и обработка результатов.

ГЛАВА 1. ОБЗОР АНАЛОГОВ . ИНСТРУМЕНТЫ РАЗРАБОТКИ

1.1 Аналоги разработки - медицинские сайты SYMPTOMATE, SYMPTOMA, LINKEMED

Среди аналогов можно выделить:

- SYMPTOMATE [1]
- SYMPTOMA [2]
- LINKEMED [3]

1.1.1 Symptomate

Symptomate.com - это удобный инструмент, созданный врачами и специалистами, который позволяет вам самостоятельно проверить свои симптомы и получить представление о возможных диагнозах.

Преимущества:

- Symptomate.com предлагает простой и понятный интерфейс для проверки симптомов и получения возможного диагноза.
- Он базируется на продвинутой системе оценки симптомов и цифровой триажной платформе.
- Пользователи могут добавлять множество симптомов и отвечать на несколько вопросов, чтобы получить наиболее вероятные условия.
- Symptomate имеет большую базу данных, накопленную благодаря более чем 90 000 часам работы врачей и проведению более 14 миллионов интервью каждый месяц.
- Он также предлагает различные уровни мед. Помощи и образовательные статьи для более полного понимаю пользователей.

1.1.2 Symptoma.ru

Symptoma.ru - это медицинская поисковая система, разработанная для помощи людям в поиске возможных причин их симптомов.

Преимущества:

- Symptoma.ru является медицинской поисковой системой, позволяющей пользователям находить возможные причины и заболевания на основе введенных симптомов.
- Он предоставляет информацию о симптомах, дифференциальных диагнозах и справочных материалах для медицинских учреждений.
- Symptoma.ru предоставляет информационно-образовательный инструмент для пользователей, помогая им расширить свои знания о медицине.

Недостаток:

- Symptoma.ru не ставит диагнозы и не дает конкретных рекомендаций по лечению.

1.1.3 Linkemed.ru

Linkemed.ru была разработана командой медицинских профессионалов и разработчиков, предлагающих онлайн-консультации с врачами. Они создали эту платформу, чтобы предоставить пользователям доступ к квалифицированным врачам, которые могут помочь им с вопросами о здоровье и предоставить конкретные рекомендации и диагнозы на основе введенных симптомов. Команда Linkemed.ru стремится обеспечить удобство и доступность медицинской помощи в онлайн-формате.

Преимущества:

- Linkemed.ru предоставляет онлайн-консультации с врачами по вопросам о здоровье и симптомах.

- Пользователи могут получить конкретные рекомендации и рекомендации по диагнозу от врачей, основываясь на введенных ими симптомах.
- Linkemed.ru также предлагает услуши медицинской поддержки и консультации в режиме реального времени

Недостаток:

- Linkemed.ru не может быть ограничен доступом в некоторых регионах или требовать оплаиу за консультации.

1.2 Обзор инструментов для разработки сайта

1.2.1 Технологии HTML5, CSS3

HTML является основным языком разметки веб-страниц, определяя их структуру и контент. Он состоит из элементов, или тегов, заключенных в угловые скобки, которые указывают браузеру, как интерпретировать содержимое. Теги HTML позволяют форматировать текст, вставлять медиа-контент, создавать таблицы, списки, ссылки и многое другое. Они могут содержать атрибуты для дополнительной информации или управления поведением элементов.

Хотя HTML не является языком программирования, он играет ключевую роль в веб-разработке, создавая основу для как простых веб-страниц, так и сложных веб-приложений. Обычно HTML используется вместе с CSS для стилизации и JavaScript для добавления интерактивности.

CSS — это язык стилей, отвечающий за оформление и визуальное представление веб-страниц. Он позволяет задавать внешний вид и расположение элементов на странице, включая цвета, шрифты, размеры и отступы. Вместе HTML и CSS работают синергетически: HTML определяет структуру, а CSS — визуальное представление, что позволяет создавать современные и привлекательные веб-сайты.

1.2.2 Библиотека React.js

React.js — это JavaScript библиотека, разработанная Facebook для создания интерфейсов веб-приложений. Основная концепция React заключается в использовании компонентов — небольших, переиспользуемых блоков интерфейса, которые могут иметь свои состояния и обновляться на их основе. Компоненты организуются в иерархию, упрощая создание сложных интерфейсов.

React использует виртуальный DOM для эффективного обновления интерфейса, что улучшает производительность приложений. Однонаправленный поток данных в React передает данные сверху вниз через пропсы (props), создавая чистые и предсказуемые компоненты, которые легче тестировать и повторно использовать.

React имеет обширную экосистему, включая библиотеки и инструменты, такие как React Router для маршрутизации и Redux для управления состоянием приложения. Благодаря своей гибкости и производительности, React стал одним из самых популярных инструментов веб-разработки.

React постоянно развивается, улучшая производительность и добавляя новые функции. Разработчики могут рассчитывать на обширную документацию и активное сообщество для решения своих задач. Кроме того, React используется для создания мобильных приложений с помощью React Native, что позволяет использовать единый код для разных платформ, экономя время и ресурсы.

1.2.3 Технология сборки приложений Vite

Vite — современная технология сборки приложений, разработанная для быстрой и эффективной веб-разработки. В отличие от традиционных инструментов, таких как Webpack или Parcel, Vite использует раздельную сборку (dev server) для разработки, что ускоряет процесс. Вместо полной сборки проекта Vite загружает модули по мере необходимости, позволяя мгновенно отображать изменения кода.

Vite обладает расширяемой архитектурой и поддерживает множество плагинов для интеграции с различными инструментами. Разработчики могут

создавать собственные плагины или использовать существующие из экосистемы Vite для расширения функциональности.

В целом, Vite предлагает быструю и эффективную разработку веб-приложений, улучшая производительность разработчиков и сокращая время сборки. Благодаря своим преимуществам и инновационному подходу, Vite становится все более популярным в сообществе разработчиков.

1.2.4 CSS-фреймворк Tailwind

Tailwind CSS — мощный и гибкий CSS-фреймворк, позволяющий создавать пользовательские интерфейсы с помощью набора переиспользуемых классов. В отличие от традиционных фреймворков, Tailwind CSS не предоставляет готовых компонентов, а предлагает низкоуровневые классы для создания стилей на лету.

Стиль Tailwind CSS основан на атомарном подходе, где каждый класс отвечает за конкретное свойство стиля. Например, вместо использования класса `.button`, можно комбинировать классы, такие как `.bg-blue-500`, `.text-white`, `.py-2`, `.px-4` для определения фона, цвета текста, отступов и размера кнопки. Это обеспечивает максимальную гибкость и контроль над стилями.

Tailwind CSS ускоряет разработку, обеспечивая контроль над стилями и облегчая создание адаптивных дизайнов, что делает его популярным среди разработчиков.

1.2.5 Библиотека Material-UI

Material-UI — популярная библиотека UI, основанная на принципах Material Design от Google. Она предоставляет готовые компоненты, такие как кнопки, поля ввода, таблицы, модальные окна и навигационные панели, обеспечивая единообразный и современный вид.

Библиотека позволяет настраивать и расширять компоненты под потребности проекта. Material-UI также предлагает удобные инструменты для работы с темами и стилями, позволяя легко настроить цветовую палитру,

типографику и отступы. Поддерживается адаптивность для различных размеров экрана, что делает Material-UI универсальным выбором для веб-разработчиков.

1.2.6 Библиотека компонентов Shadcn/ui

shadcn/ui - это не просто библиотека компонентов или UI-фреймворк. Это коллекция° переиспользуемых компонентов, которые можно копировать и вставлять в свои приложения. Основанная на Radix UI и Tailwind CSS, эта коллекция предлагает компоненты, которые отличаются красивым дизайном и доступностью.

Важно отметить, что shadcn/ui не распространяется как npm-пакет. Вместо этого вы выбираете нужные компоненты, копируете и вставляете код в свой проект, а затем настраиваете его под свои нужды. Это может служить отличной отправной точкой для создания собственной библиотеки компонентов.

1.2.7 Библиотека Zod

Zod — библиотека для валидации данных в JavaScript и TypeScript, предоставляющая простые и элегантные средства для проверки данных. Она идеально подходит для валидации пользовательских вводов, API-запросов и других сценариев.

Zod отличается интуитивно понятным API, позволяющим легко определять схемы данных и правила валидации. Она поддерживает различные типы данных, включая строки, числа, булевые значения, объекты, массивы и пользовательские типы данных. Библиотека предлагает широкий набор встроенных валидаторов для проверки обязательных полей, формата электронной почты, числовых диапазонов и других условий. Также можно создавать пользовательские валидаторы для сложных проверок.

Интеграция с TypeScript обеспечивает статическую типизацию, что позволяет проверять соответствие данных схемам на этапе компиляции, повышая безопасность кода. Zod активно поддерживается и обновляется, регулярно

добавляя новые функции и улучшения, что делает её надежным инструментом для валидации данных.

1.2.8 NodeJS и его библиотека Express

Node.js — среда выполнения JavaScript на сервере, позволяющая разработчикам выполнять JS-код на сервере, а не только в браузере. Node.js предоставляет множество возможностей для создания серверных приложений, включая обработку HTTP запросов, работу с файловой системой и базами данных.

Express.js — минималистичный и гибкий веб-фреймворк на основе Node.js. Он упрощает создание веб-приложений и API, обеспечивая удобное управление маршрутами, запросами и ответами. Express.js предлагает обширный набор функций, включая маршрутизацию, работу с сеансами и обработку ошибок.

Благодаря своей простоте и гибкости, Express.js стал одним из самых популярных веб-фреймворков для Node.js, широко используемым для создания RESTful API и веб-приложений.

1.2.9 Система управления базами данных MySQL

MySQL — популярная СУБД для хранения и управления структурированными данными в веб-разработке и других приложениях. Она обладает множеством функций, включая поддержку SQL, транзакции, множество соединений и методы аутентификации для безопасности данных. MySQL открытый исходный код с обширным сообществом разработчиков, что обеспечивает поддержку, обновления и множество расширений. Простота использования и гибкость делают MySQL популярным выбором для приложений различных масштабов, интегрируемого с другими технологиями.

ГЛАВА 2 РАЗРАБОТКА САЙТА

2.1 Формирование структуры сайта

Структура сайта для проекта по постановке диагноза на основе симптомов имеет особенности, отличающиеся от обычных веб-сайтов с разветвленной структурой. В данном случае, структура представляет собой последовательную цепочку действий, которые пользователь выполняет на сайте.

Главная страница, является отправной точкой для пользователей. С главной страницы пользователь может перейти на страницу регистрации, где ему предлагается создать учетную запись. После успешной регистрации пользователь автоматически перенаправляется на страницу опроса.

На странице опроса пользователю предлагается ответить на ряд вопросов о своем состоянии здоровья и симптомах. После прохождения опроса пользователь перенаправляется на страницу ввода симптомов, где ему предлагается ввести дополнительные сведения о своем состоянии.

После ввода симптомов происходит проверка данных, введенных пользователем, для постановки диагноза. Если достаточно данных для постановки диагноза, пользователь переходит на страницу с результатом, где ему предоставляется информация о возможном диагнозе и рекомендации.

Если данных недостаточно для постановки диагноза, пользователь перенаправляется на страницу с дополнительным тестом или опросом. После прохождения дополнительного теста пользователь снова проверяется на наличие достаточных данных для постановки диагноза и соответствующим образом перенаправляется на страницу с результатом или на страницу с дополнительным тестом.

Таким образом, структура сайта для проекта по постановке диагноза на основе симптомов представляет собой последовательную цепочку страниц и действий, которые пользователь выполняет для получения результата.

2.2 Дизайн страниц сайта

2.2.1 Главная

Почти любой сайт начинается с главного экрана, это его неотъемлемая часть. На главной странице моего сайта присутствует header с логотипом компании, а также кнопкой, при нажатии на которую мы перейдем на страницу регистрации.

Однако, самое важное на Главной странице сайта - это главный экран. Здесь представлены слова: "MedExpert - твой карманный врач". Эта фраза несет в себе обещание доступного медицинского сопровождения в онлайн-формате, способного помочь определить возможный диагноз по введенным симптомам. А рядом с этим обещанием вы обнаружите кнопку, ведущую на путь регистрации.

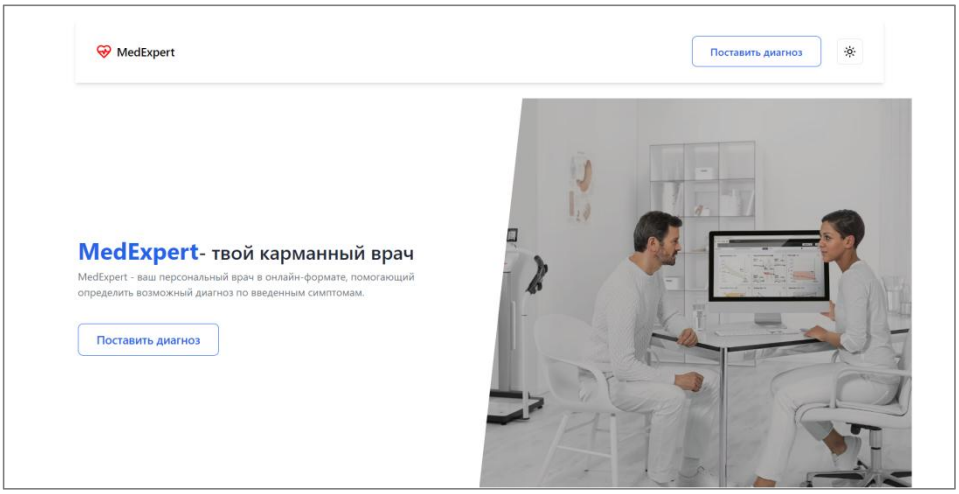

Рисунок 2.1 – Главный экран

Также внизу сайта находится footer, который содержит форму обратной связи, с помощью которой пользователи смогут связаться с администраторами сайта в случае проблем или вопросов.

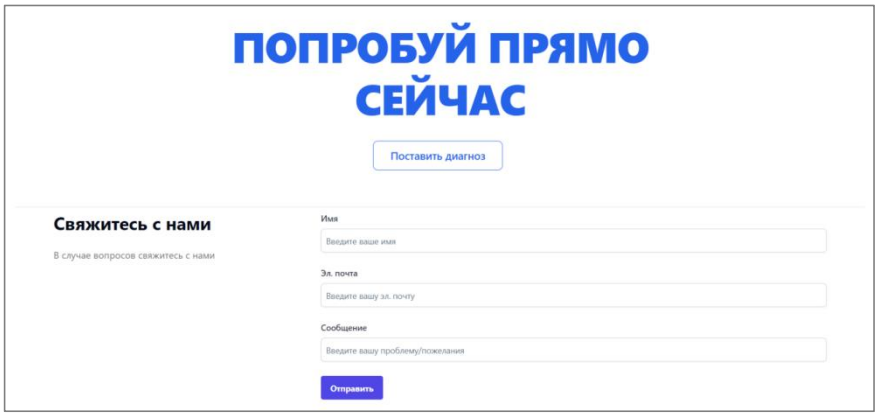

Рисунок 2.2 – Блок с призывом к действию и Footer

2.2.2 Страница “Регистрация”

Важным элементом моего сайта является страница с регистрацией пользователя. Не пройдя регистрацию пользователь не сможет перейти дальше. Страницы регистрации включает в себя краткое описание сервиса, а также форму регистрации. Сама форма содержит основные поля, такие как имя, адрес электронной почты, а также пол и возраст. Пол и возраст необходимо для последующий вычислений .

2.2.3 Страницы “Опрос/Тест”

Две страницы: Опрос и Тест являются вспомогательными, первая из них нужна для того, чтобы последующие вычисления учитывали физиологические особенности и привычки человека. Сама страница представляет собой небольшой опрос с тремя вариантами ответов: Да, Нет, Не знаю.

Страница с тестированием, нужно не всегда, пользователь переходит на нее только в случае когда не удается точно поставить диагноз. Эта страница содержит небольшой тест, вопросы в котором строятся в зависимости от проведенных ранее вычислений.

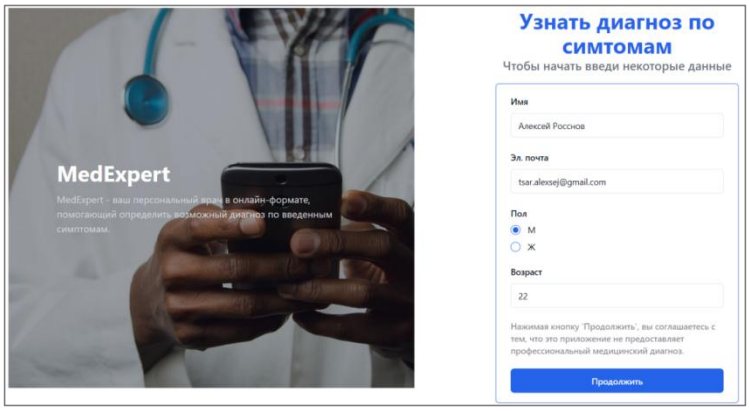

Рисунок 2.3 – Страница “Регистрация”

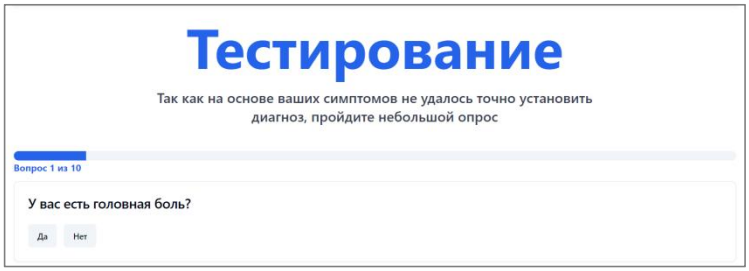

Рисунок 2.4 – Страница “Тест”

2.2.4 Страница “Ввод симптомов”

Наверное одной из самых важных страниц является страница, которая занимается вводом симптомов. Именно благодаря этой странице пользователь может получить примерный диагноз. Сама страница состоит из поля ввода и кнопки. Поле ввода содержит подсказки, которые помогут пользователю правильно написать свои проблемы, а также вспомнить, в случае если он их забыл.

2.2.5 Страница “Результат”

Завершающей страницей сайта является страница, которая показывает нам результаты нашим вычислений и показывает наиболее вероятный диагноз

пользователя. Страницы состоит из топ-3 самых вероятных диагнозов, с указанием их процента вероятности, а также дополнительной информацией, которая оформлена в виде аккордиона. В качестве дополнительной информации пользователь может узнать кртакую информацию о болезни, вопросы, которые он может узнать у врача, чтобы больше разобраться в проблеме, а также сведения о том, что делать далее. Также внизу страницы присутствует карта, на которой уже отмечены ближайшие аптеки, больницы. При нажатии на значок на карте, можно узнать подробную информацию об объекте, такие как название, время работы, адрес. Это должно помочь пользователю не тратить время на поиски ближайших медицинских учереждений.

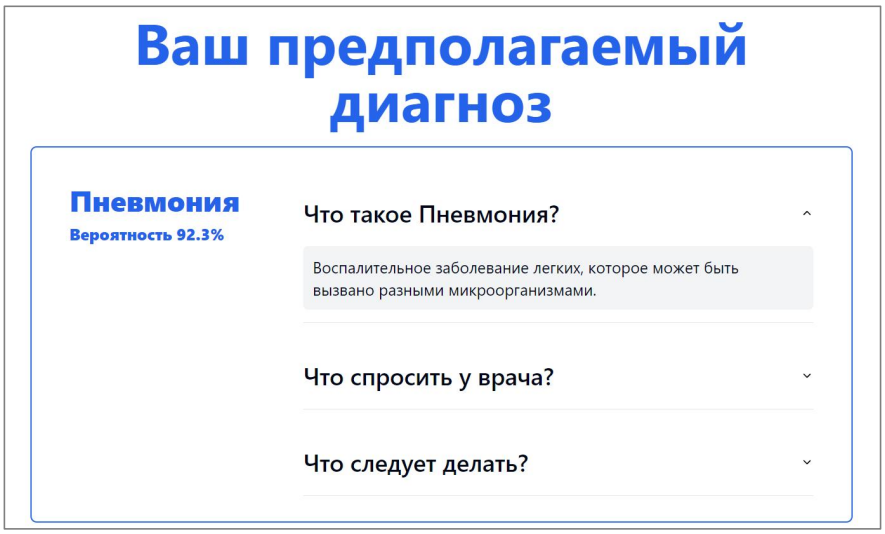

Рисунок 2.5 – Страница “Результат”

2.2.6 Страница “Ошибка 404”

Ни один сайт не может обойтись без страницы ошибки, ведь от ошибок не застрахован никто. Сама страница 404 выполнена в минималистичном стиле, впрочем как и весь сайт, и содержит заголовок, кнопку и изображение.

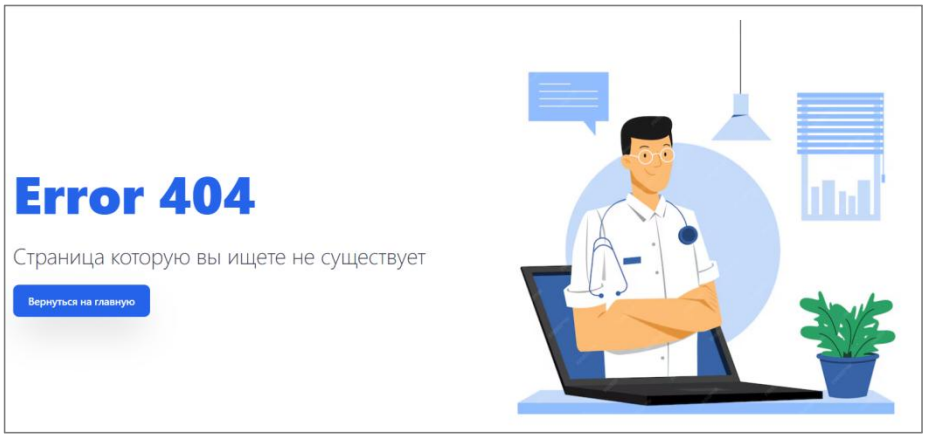

Рисунок 2.6 – Страница “Ошибка 404”

При создании дизайна страниц выбирался сдержанный и минималистичный стиль, который не будет отвлекать от главной цели. Цветовая палитра состоит всего из двух цветов: белого и синего. Белый как символ чистоты и стерильности, и синий, символизирующий надежность.

2.3 Разработка страниц сайта

Для разработки веб-сайта использовалась библиотека React.js, а также сторонние библиотеки, которые помогут быстро и просто создать удобный интерфейс.

Для создания каждой страницы на React, мы подключаем необходимые библиотеки и модули, создаем функцию, возвращающую HTML-код, и импортируем эту функцию для передачи в компоненту App.

Далее мы рассматриваем каждую страницу сайта и используем соответствующие пользовательские скрипты для их реализации.

2.3.1 Header, footer, главная

Изначально рассмотрим header и footer.Отметим, что эти объекты есть на всех страницах. Это сделано для удобство пользователя. Рассмотрим, как это было реализовано в коде:

Листинг 1. Header.

```
const Header = () => {
  return (
    <header className="py-6">
      <div className="max-w-screen-xl mx-auto px-4 flex items-center justify-between">
        <div className="□bg-white bg-opacity-5 ■text-black shadow-md hidden md:flex items-center flex-1">
          <div className="container mx-auto flex items-center h-24">
            <Link to={"/"} className="flex items-center space-x-2 mr-auto">
              <HeartPulse color="red" size={24} />
              <h4 className="font-semibold text-lg">MedExpert</h4>
            </Link>
            <div className="flex items-center space-x-4">
              <div className=" rounded-lg font-semibold ">
                <Link
                  to={"/SignIN"}
                  className="border py-3 px-7 rounded-lg border-primary text-primary mr-2 hover:bg-primary □hover:text-white transition
                  duration-300 ease-in-out"
                >
                  Поставить диагноз
                </Link>
              </div>
              <ModeToggle />
            </div>
          </div>
        </div>
      </div>
    </header>
  );
};
export default Header;
```

Как видно из изображения в коде активно используется tailwind, а также изображения иконок берутся из библиотеки Lucide-React. Эта библиотека предоставляет большое количество векторных иконок, которые можно модернизировать и изменять под свои нужды.

Что касается footer, то как говорилось ранее он содержит форму обратной связи, валидация который происходит с использованием библиотеки React - Zod. Про валидацию и форму будет рассказано далее.

И так как, главная страница не содержит интерактивных элементов то ее описание мы опустим.

2.3.2 Регистрация пользователя

Чтобы воспользоваться всем функционалом нашего сайта, пользователю необходимо ввести некоторые свои данные, а именно имя, почту, пол и возраст. Валидацией всех форм на сайте занимается библиотека React Hook Form и Zod, они следят за тем, чтобы поля не были пустыми и, чтобы пользователь корректно ввел почту.

Листинг 2. Фрагмент кода с полем ввода Имя

```
<FormField
  control={form.control}
  name="name"
  render={({ field }) => (
    <FormItem className="shad-form-item">
      <FormLabel>Имя</FormLabel>
      <FormControl>
        <Input
          type="text"
          className="shad-input"
          {...field}
        />
      </FormControl>
      <FormMessage />
    </FormItem>
  )}
/>
```

Листинг 3. Схема объекта для валидации формы

```
import { z } from "zod";

export const SignInValidation = z.object({
  name: z
    .string()
    .min(2, { message: "Имя должно содержать не менее 2 символов" })
    .max(50, { message: "Имя должно содержать не более 50 символов" }),
  email: z
    .string()
    .email({ message: "Введите корректный адрес электронной почты" }),
  gender: z.enum(["М", "Ж"], { message: "Пожалуйста, выберите М или Ж" }),
  age: z
    .number()
    .min(16, { message: "Возраст должен быть не менее 16 лет" })
    .max(99, { message: "Возраст должен быть не более 99 лет" }),
});
```

В случае ввода некорректных данных, неверное поле формы окрасится в красный цвет, а также появится сообщение,в котором будет описана проблема.

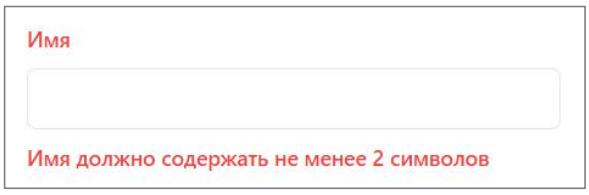

Рисунок 2.7 – Некорректные данные поля Имя

В случае, если пользователь правильно введет свои данные, он сможет перейти дальше, а егго данные отправяться на сервер.

Листинг 4. Функция, отправляющая данные клиента на сервер

```
async function onSubmit(values: z.infer<typeof SignInValidation>) {
  setIsLoading(true);
  try {
    const response = await axios.post("http://localhost:5000/client", values);
    console.log(response.data);
    navigate("/Survey");
  } catch (error) {
    console.error("Ошибка при отправке данных:", error);
  }
  setIsLoading(false);
}
```

2.3.3 Ввод симптомов

После перехода на страницу ввода симптомов, пользователь сталкивается с основополагающим этапом процесса постановки диагноза. Здесь представлено поле для ввода, которое разработано с использованием Material UI, а именно с помощью тега Autocomplete, встроенного в эту библиотеку.

Важно отметить, что часто у пользователя возникает не один симптом, а несколько. Поэтому поле для ввода сделано множественным (multiply), чтобы пользователь мог выбирать несколько симптомов одновременно. Это значительно упрощает и ускоряет процесс ввода информации.

Кроме того, стоит упомянуть о подсказках при вводе. При вводе своих симптомов пользователю предоставляется либо весь список симптомов для выбора, либо, если пользователь вводит симптомы вручную, поле автоматически отображает совпадения по подстроке. Это делает процесс ввода более удобным и эффективным.

Данные о симптомах получаются с сервера с помощью кастомного хука useFetch, который обеспечивает обмен данными между клиентом и сервером, обеспечивая плавную и быструю загрузку информации.

Листинг 5. Поле для ввода

```
return (
  <Stack spacing={2} width="800px">
    <Autocomplete
      multiple
      id="tags-standard"
      options={Symptoms}
      getOptionLabel={(option) => option.SymptomsName}
      onChange={handleChange}
      noOptionsText="Нет доступных вариантов"
      isOptionEqualToValue={(option, value) =>
        option.SymptomsID === value.SymptomsID
      }
      renderInput={(params) => (
        <TextField
          {...params}
          variant="standard"
          label="Симптомы"
          placeholder="Введите ваши симптомы"
        />
      )}
    />
  </Stack>
);
```

Кроме того, на странице ввода симптомов реализована проверка минимального количества введенных симптомов. В данном случае установлено, что пользователь должен ввести минимум 3 симптома. Это позволяет сделать вычисления более точными и увеличивает вероятность корректного вывода результата.

Проверка минимального количества симптомов является важным этапом в процессе постановки диагноза. Она обеспечивает более надежные и достоверные результаты, поскольку чем больше симптомов введено пользователем, тем более детальную картину состояния здоровья можно получить. Такой подход способствует повышению качества диагностики и, соответственно, улучшению общего опыта пользователей.

2.3.4 Страница “Результат”

Итак, ключевой страницей, ради которой пользователь проходит весь процесс ввода данных и симптомов, является страница с результатом. Эта страница является завершением всего сайта.

На странице результата, помимо заголовка, размещены три аккордеона, которые содержат всю интересующую информацию и процент вероятности постановки диагноза. Аккордеоны организованы таким образом, чтобы пользователь мог легко и быстро найти нужные данные:

Определение диагноза: В этом аккордеоне содержится подробное описание предполагаемого диагноза, включая основные симптомы, возможные причины и характерные особенности заболевания. Это помогает пользователю понять природу своего состояния и получить базовые знания о диагнозе.

Вопросы для врача: Этот аккордеон содержит список вопросов, которые пользователь может задать врачу на приеме. Вопросы подобраны таким образом, чтобы помочь пользователю получить максимально полную информацию о своем состоянии, возможных вариантах лечения и прогнозах. Это способствует более продуктивному взаимодействию с медицинским специалистом.

Алгоритм действий: Здесь представлен пошаговый план действий, который следует предпринять после постановки диагноза. В алгоритме указаны рекомендации по лечению, включая медицинские препараты, список которых меняется в зависимости от диагноза. Этот раздел помогает пользователю сориентироваться в дальнейших шагах и понять, какие меры необходимо предпринять для улучшения своего состояния.

В завершение страницы расположена интерактивная карта, которая позволяет пользователю легко найти ближайшие аптеки или больницы. При нажатии на маркировку на карте пользователь получает подробную информацию о местоположении, названии и времени работы учреждения. Это значительно упрощает процесс поиска медицинской помощи и лекарственных препаратов, обеспечивая пользователю доступ к необходимым ресурсам в непосредственной близости.

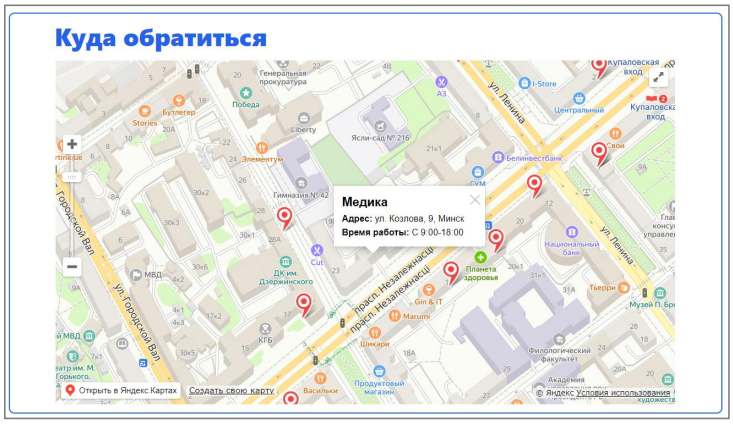

Рисунок 2.8 – Интерактивная карта

Страница результата не только предоставляет подробную информацию о диагнозе и дальнейших шагах, но и делает процесс получения этой информации удобным и понятным для пользователя. Благодаря тщательно продуманной структуре и полезным функциональным элементам, страница результата становится ценным инструментом для пользователей, стремящихся улучшить свое здоровье и получить качественную медицинскую помощь.

Листинг 6. Аккордион

```
<Accordion type="single" className="mb-8" collapsible>
  <AccordionItem value="item-1">
    <AccordionTrigger className="text-3xl font-semibold mb-2">
      Что такое {name}?
    </AccordionTrigger>
    <AccordionContent>
      <p className=" bg-gray-100 p-3 text-xl rounded-md">
        {description}
      </p>
    </AccordionContent>
  </AccordionItem>
</Accordion>
```

Также хочется отметить , что сам аккордион сделан при помощи библиотеки переиспользуемых компонентов shadcn/ui. Эта библиоотека только развивается, но уже имеет хороший набор компонентов. Причем каждый из компонентов можно изменять под себя.

2.3.5 Страница с тестом

Страница теста – страница, где пользователь должен пройти тест для получение больше четкой картины о его диагнозе. Страница содержит 10 вопросов, которые генерируются на основе проведенных ранее вычислений. Также для удобства пользователя был добавлен шкала с прогрессом, Сама шкала была сделана при помощи тега Progress из shadcn/ui. На эту страницу попадает не все пользователи, а лишь те, данных о симптомах которых не хватает для постановки диагноза. Это достигается путем установления минимального порогового значения(по умолчанию это значение стоит на 80%) для процента вероятности диагноза. Если вычисления показали, что процент меньше порогового, то пользователь переходит на страницу с тестом, в противном же случае переход осуществляется на страницу с результатами.

2.3.6 Страница с опросом

Это важный этап, на который пользователь переходит сразу после регистрации, и он является обязательным для продолжения. Проходя этот опрос, пользователь предоставляет больше данных о своем здоровье и физических особенностях. Эти дополнительные данные играют ключевую роль в улучшении точности диагностики, так как определенные физиологические особенности могут либо предотвратить, либо повысить риск возникновения определенных диагнозов.

Страница с опросом тщательно разработана для удобства пользователя. Вопросы сформулированы так, чтобы собрать наиболее важную информацию без перегрузки пользователя, она содержит лишь 5 вопросов.

2.3.7 Страница 404

Данная страница предназначена для тех случаев, когда пользователь перешел по неправильной ссылке. Попав на эту страницу, пользователю предлагается вернуться на главную.

2.4 Разработка алгоритма

Компьютерная диагностика заболеваний является для врача таким же важным инструментом, как расчеты для инженера. Она не заменяет врача, а лишь помогает ему в работе. Поэтому важно развивать математические методы диагностики и оценивать их эффективность.

Постановка диагноза врачом – результат сложного процесса осмысливания и сопоставления большого количества информации. Достоверность диагноза зависит от величины накопленного опыта в данной области медицины. Результаты диагностического процесса зависят также от личных качеств врача. Методы автоматической диагностики свободны от многих свойственных человеку недостатков, но уступают человеку в области творчества, интуиции. В настоящий момент для постановки диагноза с помощью компьютера используются различные методы: вероятностные, обучения распознаванию, математической логики и др. В

данной курсовой работе рассматривается вариант реализации алгоритма для поддержки принятия решений на основе метода Байеса.

В своем выборе я ссылался на статью, опубликованную в научном журнале “ПОЛЗУНОВСКИЙ ВЕСТНИК” в которой проводились исследования. В статье опубликована таблица с результатами их исследований [11]:

Таблица 1

Метод диагностики и процедура оценки точности	Процент поставленных диагнозов		
	Правильных	Неправил.	Неопределен.
1а. Дискриминантный анализ	91	9	0
2а. Деревья классификации	90,5	9,5	0
3а. Формула Байеса	85,5	2	12,5
4а. Нейронные сети	90	5,5	4,5

Исходя из таблицы формула Байеса показывает лучшие результаты, по количеству неправильно-поставленных диагнозов, то есть вероятность того, что диагноз будет поставлен неправильно сведен к минимума. Главной проблемой метода как видно из таблицы является неопределенность, то есть процент нескольких диагнозов схож. Это проблема будет решена далее.

Также выбор пал именно на формулу Байеса из-за его того, что он эффективно сочетает априорные вероятности с новыми данными, что обеспечивает более точные и надежные результаты. Благодаря своей гибкости и математической обоснованности, метод Байеса является мощным инструментом для медицинской диагностики, что делает его идеальным выбором для данной работы.

Перейдем с рассмотрению алгоритма.

Теорема Байеса многим известна из теории вероятностей, и записывается в виде формулы (1):

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \quad (1)$$

Где P(A) - априорная вероятность гипотезы A (заранее изветсная вероятность)

P(A|B) - вероятность гипотезы A при наступлении события B (апостериорная вероятность)

P(B|A) — вероятность наступления события B при истинности гипотезы A

P(B) — полная вероятность наступления события B.

Но заболеваний много и симптомов тоже, поэтому эту формулу следует адаптировать под наши требования.

Пусть требуется провести дифференциальную диагностику между заболеваниями D_1, D_2, D_3,… D_n. Каждое из этих заболеваний характеризуется распределением условных вероятностей $P(S \mid D_j)$ появления у пациента того или иного комплекса симптомов. Пусть $S=\{S_1 \ ,...,S_k \ ,...,S_n\}$, где S_i — возможные значения (градации) различных симптомов. Если бы эти распределения, а также априорные вероятности заболеваний $P(D_j)$ были известны, задача дифференциальной диагностики свелась бы к статистической задаче выбора гипотез. Оптимальное диагностическое правило для этой задачи можно построить с использованием известной формулы Байеса. Последняя для апостериорной вероятности диагноза D_j дает выражение:

$$P(D_j \mid S_i) = \frac{P(D_j)P(S|D_j)}{\sum_j P(D_j)P(S|D_j)}, \qquad (2)$$

где $P(D_j)$ - априорная вероятность заболевания с диагнозом D_j среди рассматриваемой группы болезней; $P(S|D_j)$ - вероятность появления комплекса признаков при диагнозе D_j .

Существенным препятствием к широкому распространению такого решающего правила является отсутствие заданных распределений ($P(S|D_j)$ и вероятностей $P(D_j)$. Вероятность заболеваний ($P(D_j)$ можно достаточно точно оценить, вычислив частоты на большом клиническом материале. Однако определение распределения $P(S|D_j)$ значительно сложнее, так как для каждого диагноза необходимо определить условную вероятность любой комбинации признаков.

Для 30 двоичных клинических параметров существует свыше миллиарда возможных комбинаций симптомов. Очевидно, собрать достаточный клинический массив для оценки $P(S|D_j)$ невозможно. Единственным выходом является использование аппроксимации вместо точного значения $P(S|D_j)$. Вероятностные алгоритмы различаются по способу аппроксимации $P(S|D_j)$.

Наиболее распространено предположение о статистической независимости симптомов у больных. Тогда мы получим следующее:

$$P(S \mid D_j) = \prod_i P(S_i \mid D_j)\,. \tag{3}$$

Величина $P(S_i|D_j)$ определяется на основании данных медицинской статистики, результатов обработки архивного материала и литературных данных, причем она будет тем более достоверна, чем больше n.

Но также рассмотрим случай отсутствия какого-либо симптопа. Вероятность отсутствия симптома равна:

$$P(\overline{S}_1 \mid D_j) = 1 - P(S_1 \mid D_j) \tag{4}$$

Расчет проводится точно таким же образом, но вероятность $P(S_i|D_j)$ в заменяется на $1\text{-}P(S_i|D_j)$.

Полученные вероятности сравниваются с определенной пороговой величиной T_j, которая устанавливается в процессе обучения системы. Если $P(D_j) > T_j$, то делается вывод о наличии у пациента диагноза D_j. В моей курсовой работе это значение установлено на 80% В случае, когда $P(D_j) < T_j$, необходимо проведение дополнительных обследований.

В моей работе для повышения точности вычислений я использовал принцип "Динамических априорных вероятностей", где априорные вероятности диагнозов изменяются в зависимости от физиологических особенностей пользователя.

На этапе регистрации пользователь вводит свой пол и возраст. Например, у пожилых людей вероятность заболеть гриппом выше. Эта информация помогает скорректировать априорные вероятности диагнозов, делая их более точными.

Вторым этапом является прохождение опроса, включающего вопросы типа "Я курю сигареты" и "У меня снижена иммунная система". Эти данные дополнительно корректируют вероятности диагнозов, аналогично информации о поле и возрасте.

Также на сайте предсмотрено тестирование. Если по ходу вычисления окажется, что результат меньше минмального значения, то пользователь переходит на этап тестирования, проходя тест пользователь уточняет свои данные.

Принцип "Динамических априорных вероятностей" позволяет учитывать индивидуальные особенности каждого пользователя, что повышает точность диагностики и делает систему более адаптивной и персонализированной.

Все это позволило создать алгоритм, который может достаточно точно поставить диагноз пользователю, учитывая его особенности организма.

2.5 Серверная часть

В курсовом проекте реализована серверная часть с использованием Node.js и фреймворка Express, которая взаимодействует с базой данных MySQL.

2.5.1 Подключение к базе данных. Запросы к базе данных.

В проекте для взаимодействия с базой данных MySQL используется библиотека mysql для Node.js. Ниже приведен пример кода, который показывает, как настроить подключение к базе данных.

Листинг 7. Подключение к базе данных

```
const db = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "MedSystem",
});
```

В данном фрагменте кода:host: Адрес сервера базы данных. В данном случае используется localhost, что означает, что база данных размещена на том же компьютере, что и серверное приложение;user: Имя пользователя базы данных. Здесь используется пользователь root; password: Пароль пользователя базы данных. В примере пароль пустой, но в реальных приложениях рекомендуется использовать сложные пароли; database: Имя базы данных, к которой подключается приложение. В данном случае это MedSystem.

Далее устанавливается соединение с базой данных.

Основные запросы к базе данных - это получение данных с нее , а также запись туда.

Листинг 8. Запрос на запись клиента в базу данных

```
app.post("/client", (req, res) => {
  const q =
    "INSERT INTO client(`name`, `email`, `age`, `gender`) VALUES (?, ?, ?, ?)";
  const values = [req.body.name, req.body.email, req.body.age, req.body.gender];

  db.query(q, values, (err, data) => {
    if (err) return res.json(err);
    return res.json("User has been added");
  });
});
```

Листинг 9. Запрос на получение данных с сервера

```
app.get("/Diagnoses", (req, res) => {
  const q = "SELECT * FROM `Diagnoses`";
  db.query(q, (err, data) => {
    if (err) return res.json(err);
    return res.json(data);
  });
});
```

2.5.2 База данных

Схема базы данных представляет собой набор связанных таблиц, каждая из которых содержит информацию, необходимую для функционирования медицинской системы.

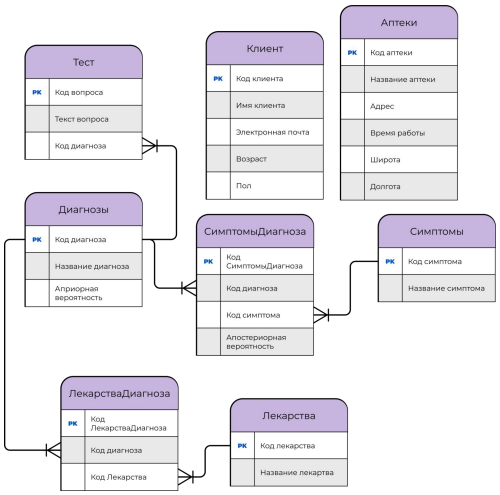

Рисунок 2.9 – Схема базы данных

В самой схеме присутствуют: таблица клиент, которая хранит данные о пользователях; таблица аптек, которая хранит адреса и расположение на карте самих аптек, что нужно для создание карты, таблица диагнозов и симптом, а также таблица разрешающая связь многие ко многим СимптомыДиагноза. И аналогично для таблиц диагноз и лекартсво. Также в БД присутствует таблица с тестом.

Схема базы данных включает восемь таблиц, каждая из которых выполняет определенную роль в медицинской системе. Таблицы связаны между собой с помощью внешних ключей, что позволяет эффективно хранить и обрабатывать данные о клиентах, диагнозах, симптомах, лекарствах и аптеках. Такая структура обеспечивает гибкость и масштабируемость системы, позволяя легко добавлять новые функции и интегрировать дополнительные источники данных.

2.5.3 Реализация рассылки.

Далее рассмотрим реализацию функции рассылки с электронной почты с использованием библиотеки nodemailer. Основная задача заключается в отправке электронных писем пользователям с полезной информацией, а также рекламой.

Первым этапом является установка соответствующих библиотек и их подключение. Для реализации функции рассылки используется библиотека nodemailer, которая позволяет отправлять электронные письма из Node.js приложения. Также используется пакет dotenv для безопасного хранения конфиденциальных данных, таких как адрес электронной почты и пароль приложения.

Транспортер — это объект, который отвечает за отправку электронной почты. В данном проекте используется SMTP-сервер Outlook. Для настройки транспортера указываются параметры подключения, такие как хост, порт и учетные данные.

Листинг 10. Транспортер

```
const transporter = nodemailer.createTransport({
  host: "smtp.office365.com", // Сервер Outlook для SMTP
  port: 587, // Порт для SMTP сервера Outlook
  secure: false,
  auth: {
    user: process.env.USER,
    pass: process.env.APP_PASSWORD,
  },
});
```

После этого создаются параметры для самого письма. Параметры письма включают отправителя, получателей, тему и содержимое письма.

Для отправки письма используется функция sendMail, которая принимает транспортер и параметры письма. В случае успешной отправки в консоли выводится сообщение "Письмо отправлено". В случае ошибки она также выводится в консоль.

Листинг 10. Отправка письма

```
const sendMail = async (transporter, mailOptions) => {
  try {
    await transporter.sendMail(mailOptions);
    console.log("Письмо отправлено");
  } catch (error) {
    console.log(error);
  }
};

sendMail(transporter, mailOptions);
```

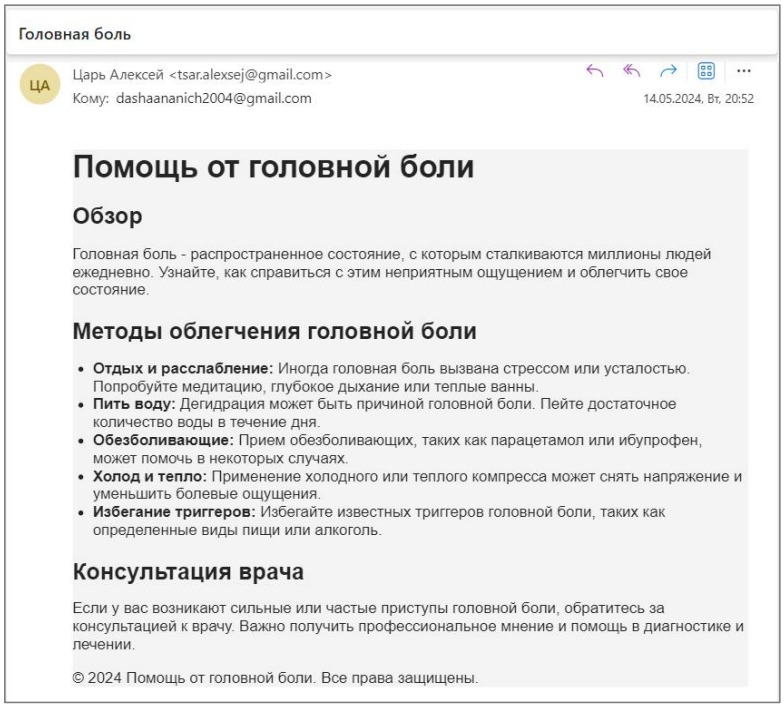

Рисунок 2.10 – Пример письма

Реализация функции рассылки электронной почты с использованием nodemailer позволяет отправлять информативные письма пользователям. Это может быть полезно для различных целей, таких как уведомления, рассылка новостей или информационные бюллетени.

ЗАКЛЮЧЕНИЕ

В результате написания курсовой работы были достигнуты все поставленные задачи. Было разработано семь страниц сайта, включая главную, страницу ввода симптомов, регистрацию, тестирование, опрос, результаты и страницу 404.

Для создания дизайна страниц были использованы современные библиотеки, такие как Tailwind CSS, Material-UI и Shadcn/UI, что обеспечило эстетическое и функциональное оформление сайта. Кроме того, была разработана интерактивная карта для удобства пользователей.

Для хранения всей необходимой информации, включая данные пользователей для последующей рассылки писем, была создана база данных. Также была разработана форма обратной связи, обеспечивающая взаимодействие с пользователями.

Веб-сайт полностью реализован, обладает красивым дизайном. Кроме того, разработ алгоритм, основанный на методе Байеса, для постановки диагноза.

Помимо этого, приложение было подвергнуто тщательному тестированию, включая различные сценарии использования. Мы рассмотрели случаи с минимальным количеством введенных симптомов, а также экстремальные ситуации с абсолютно разными симптомами. Большая часть экспериментов включала случаи, когда к симптомам диагноза добавлялся один лишний симптом с вероятностью 30%, или наоборот, один симптом убирался. Результаты тестирования показали, что приложение демонстрирует неплохие результаты, успешно обрабатывая разнообразные варианты ввода и предоставляя точные диагнозы с высокой достоверностью.

СПИСОК ИСТОЧНИКОВ

1. Symptomate [Электронный ресурс] – Режим доступа: https://symptomate.com/ – Дата доступа: 12.02.2024
2. Linkemed [Электронный ресурс] – Режим доступа: https://linkemed.ru/ – Дата доступа: 12.02.2024
3. Symptoma [Электронный ресурс] – Режим доступа: https://www.symptoma.ru/ – Дата доступа: 12.02.2024
4. Документация MaterialUI. [Электронный ресурс] – Режим доступа: https://mui.com/material-ui/getting-started/overview/ – Дата доступа: 14.04.2024
5. Документация к JavaScript-библиотеке React. [Электронный ресурс] – Режим доступа: https://react.dev/ – Дата доступа: 10.03.2024.
6. Учебник Express [Электронный ресурс] – Режим доступа: https://code.mu/ru/javascript/nodejs/book/express/ – Дата доступа: 01.04.2024.
7. Документация Zod. [Электронный ресурс] - Режим доступа: https://zod.dev/ – Дата доступа: 20.03.2024
8. Документация React-yandex-maps. [Электронный ресурс] - Режим доступа: https://pbe-react-yandex-maps.vercel.app/en/ – Дата доступа: 05.05.2024
9. Документация Shadcn/ui. [Электронный ресурс] - Режим доступа: https://ui.shadcn.com/ – Дата доступа: 01.04.2024
10. Байесовский вывод. [Электронный ресурс] - Режим доступа: https://science.fandom.com/ru/wiki/Байесовский_вывод/ – Дата доступа: 11.02.2024
11. Жмудяк М.Л., Повалихин А.Н. Автоматизированная система медицинской диагностики заболеваний с учетом их динамики.[Электронный ресурс] // «Ползуновский вестник». – №1, 2006. – С. 94-106. Режим доступа: https://journal.altstu.ru/media/f/old2/pv2006_01/pdf/095jmudyak.pdf – Дата доступа: 10.02.2024