

Real-time Healthcare Analytics

A Project Report

Presented to

The Faculty of the College of Engineering

San Jose State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science in Software Engineering

By

Preethi Billa

Thanuj kumar Janugani

Mounica Reddy Kandi

Varun Teja Maguluri

December 2023

Copyright © 2023

Preethi Billa

Thanuj kumar Janugani

Mounica Reddy Kandi

Varun Teja Maguluri

ALL RIGHTS RESERVED

DocuSigned by:
Weider Yu
597AACDDDEE9427...

[Weider Yu], Project Advisor

[Dan Harkey], Director, MS Software Engineering

[Department Chair's Name], Department Chair

ABSTRACT

Real-time Healthcare Analytics

By

Preethi Billa, Thanuj kumar Janugani, Mounica Reddy Kandi, Varun Teja Maguluri

Health prediction analysis using big data is a rapidly growing field that uses large amounts of data to predict and prevent potential health issues. This analysis can identify patterns and trends in health data, which can help to detect diseases early and develop preventative measures. Machine learning and data mining are used to analyze the data and make predictions.

One of the key benefits of using big data in health prediction analysis is that it can identify patterns and trends that may not be visible with traditional methods. For example, data from electronic health records, social media, and other sources can be analyzed to identify specific disease risk factors and develop targeted prevention strategies. Additionally, the use of machine learning and data mining techniques allows for the analysis of enormous amounts of data, which can improve the accuracy of predictions and allow for the development of personalized treatment plans.

We are aware that the entire healthcare system is built on assumptions, which must first be put to the test and proven by several tests before patients can have confidence in their doctors' expertise. To use the system's analysis to predict a person's health, we developed a system that uses data mining techniques to predict a person's health based on the results of numerous medical tests. We used the Statlog (Heart) Data Set from the UCI Machine Learning Repository to train the system. This data set contains parameters such as age, sex, type of chest discomfort, cholesterol, sugar, and results. We only need to pass a few generic inputs to make a prediction. The prediction results from all the algorithms are combined by calculating their mean value, which represents the actual result of the prediction process, which runs entirely in the background.

Health prediction analysis using big data is a promising field with the potential to improve the quality of healthcare. By identifying patterns and trends in health data, this analysis can help to detect diseases early and develop preventative measures. Additionally, the use of machine learning and data mining techniques can improve the accuracy of predictions and allow for the development of personalized treatment plans.

Acknowledgments

The authors are deeply indebted to Professor Weider Yu for his invaluable comments and assistance in the preparation of this study.

Table of Contents

Chapter 1. Project Overview	1
1.1 Introduction	1
1.2 Proposed Areas of Study and Academic Contribution	1
1.3 Current State of Art	2
Chapter 2. Project Architecture	4
2.1 Architecture	4
2.2 Architecture Subsystems	5
2.2.1 Front-end Architecture	5
2.2.2 Backend Architecture	5
2.2.3 Deployment Architecture	7
Chapter 3. Technology Descriptions	8
3.1 Frames and Libraries	8
3.2 Databases	10
3.3 Tools and Platforms	11
3.4 Programming Language	12
3.5 Machine Learning Algorithm	12
Chapter 4. Project Design	13
4.1 Client Design	13
4.2 UI Mockups	15
4.3 Backend Design	19
4.4 Database Design	21
Chapter 5. Project Implementation	24
5.1 Client Implementation	24
5.2 Mid-tier Implementation	24
Chapter 6. Testing and Verification	27
6.1 Unit Testing	27
6.2 Integration Testing	27
6.3 Load Testing	28
Chapter 7. Performance and Benchmarks	29
7.1 Performance Metrics	29
7.2 Load Testing and Scalability	30
7.3 Benchmarking	30
Chapter 8. Deployment, Operations, Maintenance	31
Chapter 9. Summary, Conclusions, and Recommendations	32

List of Figures

Figure 1: Architecture Diagram of Healthcare Analysis	4
Figure 2: Sign up page	15
Figure 3: Login page	15
Figure 4: Home page	16
Figure 5: Heart Disease Death Rates - USA	18
Figure 6: Track your health page	20
Figure 7: Heart Health Data page	21
Figure 8: Profile page	21
Figure 9: BMI calculator page	22
Figure 10: Patient Heart Health Data page	23
Figure 11: Doctor's profile page	23
Figure 12: Authentication Flow Diagram	25
Figure 13: DB Schema	27
Figure 14 : Backend Server Deployment - Replit	37
Figure 15 : Database - SQL Workbench	38
Figure 16 : Relational Database System - AWS	38

List of Tables

Error! No table of figures entries found.

Chapter 1. Project Overview

1.1 Introduction

The main goal of this project is to develop a data-driven solution for predicting heart problems in individuals with the aim of improving the accuracy and efficiency of health prediction. Existing healthcare systems rely heavily on the expertise of healthcare professionals, which can be limited by available information and extensive experience. To overcome these limitations, the project will use data mining techniques and algorithms.

The system is designed to be easy to use and requires only a few common inputs to create forecasts. This project uses the Statlog (Heart) dataset from the UCI Machine Learning Repository as a source of truth for training the system. The dataset contains various attributes such as age, gender, type of chest pain, cholesterol, sugar, and score that can be used to train the system and predict heart problems.

The significance of this project is that it has the potential to transform the healthcare industry by reducing reliance on medical professionals and providing more accurate and efficient health predictions. The system can detect potential health problems early, so individuals can take proactive steps to improve their health and reduce their risk of heart disease.

Finally, the project rationale emphasizes the importance of providing accurate and efficient health predictions for people with heart problems. This project aims to improve the health system's reliance on assumptions and increase the accuracy and efficiency of health prediction by using data mining techniques and machine learning algorithms.

1.2 Proposed Areas of Study and Academic Contribution

"Real-time Healthcare Analytics" is a project that aims to use big data analytics and machine learning techniques to develop algorithms for early detection and prevention of heart disease. This project proposes to explore the effectiveness of various data collection

and analysis methods, such as probabilistic data collection and correlation analysis, for predicting future health conditions of patients. The role of natural language processing (NLP) in analyzing and processing big healthcare data, particularly in clinical decision support, clinical operations optimization, and cost-cutting, will also be evaluated.

In addition, the project proposes to investigate the use of Apache Spark as a large-scale distributed computing platform for real-time streaming data events in healthcare analytics. The practical benefits of big data analytics in the healthcare industry will be assessed, including the identification of practical barriers to adoption. Different approaches to handling different types of medical data, such as structured, unstructured, and semi-structured data, in big data analytics, will also be compared.

This project will develop new techniques and methodologies for using big data analytics and machine learning in the early detection and prevention of heart disease. This project will also contribute to the advancement of the understanding of the role of NLP in healthcare analytics, the use of distributed computing platforms like Apache Spark for real-time streaming data events, and the challenges associated with the handling of different types of medical data in big data analytics.

Overall, "Real-time Healthcare Analytics" will make a significant contribution to the healthcare industry by providing a robust and reliable system for early detection and prevention of heart disease. The project will also advance our understanding of the practical benefits of big data analytics and machine learning techniques in healthcare and help address the barriers to adoption of these technologies.

1.3 Current State of Art

Real-time healthcare analytics is a rapidly growing field that has the potential to revolutionize healthcare delivery. The current state of the art on real-time healthcare analytics involves the use of advanced technologies such as artificial intelligence (AI),

machine learning (ML), and big data analytics to process large volumes of healthcare data in real-time.

One of the main applications of real-time healthcare analytics is predictive analytics, which involves using historical data to predict future health outcomes. For example, predictive analytics can be used to identify patients who are at high risk of developing certain health conditions, such as diabetes or heart disease. This information can be used to develop targeted interventions that can help prevent these conditions from developing or worsening.

Real-time healthcare analytics is also being used to improve patient outcomes by providing real-time insights into patient health status. For example, wearable devices such as fitness trackers and smartwatches can be used to monitor vital signs and other health metrics in real-time. This information can be used to detect early signs of health problems and intervene before they become more serious.

Another important application of real-time healthcare analytics is in the field of population health management. By analyzing large volumes of data from diverse sources, healthcare organizations can identify patterns and trends in disease prevalence and healthcare utilization. This information can be used to develop targeted interventions that can improve health outcomes and reduce healthcare costs.

In summary, the current state of the art on real-time healthcare analytics is focused on using advanced technologies to process large volumes of healthcare data in real-time. By doing so, healthcare organizations can gain valuable insights into patient health outcomes and improve the quality of care they provide.

Chapter 2. Project Architecture

2.1 Architecture

The healthcare analytics system has an advanced and well-integrated architecture. In order to ensure a responsive and user-friendly experience for patients and healthcare providers, the user interface was first designed using React.js and Node.js. Machine learning for health analysis and prediction is provided by Scikit-Learn, and Flask-powered Python back-end handles API services. The MySQL database managed by XAMPP enables safe data storage and retrieval.

The entire system is designed to collect, process, and distribute medical data in an efficient manner. Machine learning can be used by healthcare professionals to produce insights that may be useful in making informed decisions on patient care. This architecture ensures a smooth user experience while protecting the security and integrity of healthcare data.

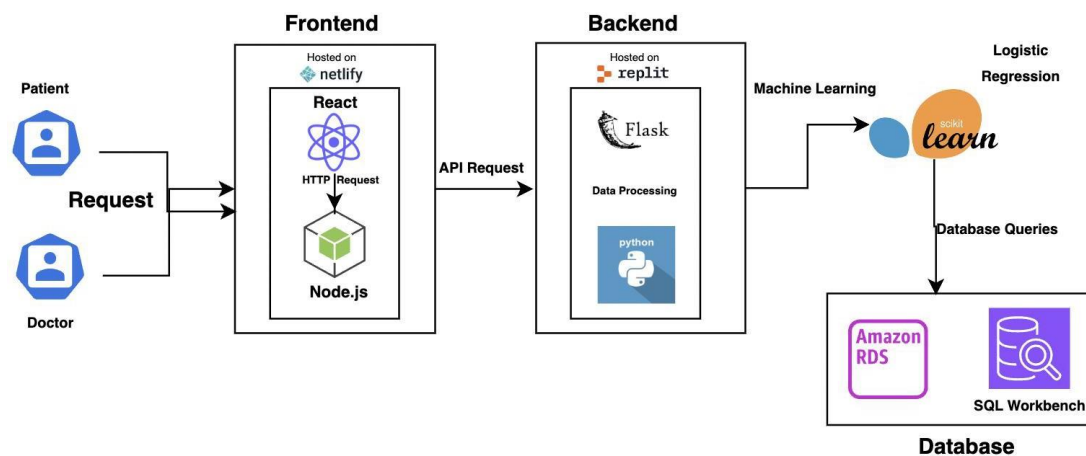


Figure 1: Architecture Diagram of Healthcare Analysis.

2.2 Architecture Subsystems

2.2.1 Front-end Architecture

The front-end of the Real-time Healthcare Analytics is developed using React JS and Node.js server.

User Interface (React.js): A JavaScript package called React.js is utilized to create the application's user interface (UI) for healthcare analytics. You can use it to make dynamic and interactive UI elements. In this instance, it's utilized to create and render the application's front end, giving patients and medical professionals an intuitive user experience.

Node.js Server: This application's front-end runs on a Node.js backend. It manages routing, client-side queries, and user service for the React.js application. Because Node.js is capable of efficiently managing several concurrent connections, its non-blocking I/O nature makes it a great choice for real-time applications.

Front-End Server (Node.js): In addition to hosting the React.js application, this server is in charge of providing users' web browsers with front-end content. In order to retrieve data or start operations, it interacts with the back end and manages client-side requests from the user interface.

2.2.2 Backend Architecture

The backend architecture acts as a middleware between the database and the frontend of the application.

APIs (Flask): Flask is the framework used to establish APIs, and Python is utilized to build the healthcare analytics system's back end. Flask is a web framework that is lightweight and adaptable that lets you create and provide RESTful APIs. These APIs offer endpoints for a number of functions, such as data retrieval, login, user registration, and heart health analysis. It serves as a link between the data processing and front-end elements.

Machine Learning (Scikit-Learn): For machine learning applications, the Python library Scikit-Learn is utilized. Healthcare data analysis and prediction are

accomplished by the application of the Logistic Regression algorithm in the design. Based on patient data, it can be used, for instance, to forecast the possibility of heart health problems.

Data Access (MySQL): User profiles, heart health data, and authentication details are among the kinds of data stored in a XAMPP-managed MySQL database. This database is accessed by Python through the MySQL connection. Web applications frequently use MySQL, a relational database management system, to store and retrieve data.

Back-End Database Server (XAMPP MySQL): The database component of XAMPP, an open-source web server solution, is MySQL. Data management and archiving is done in this configuration for user profiles, medical records, and other application data.

Additional Python Packages:

- **NumPy and Pandas:** Python libraries used for data analysis and manipulation include NumPy and Pandas. They make processes like cleansing, transformation, and filtering of data possible.
- **gdown:** Files or datasets can be downloaded using this program. When you need to obtain external datasets for analysis or training, it might be useful.
- **pickle:** Data can be serialized and deserialized using the Pickle Python package. It enables the binary loading and saving of data, including machine learning models.
- **logging:** The purpose of implementing logging is to track application events and produce logs. It facilitates debugging and behavior monitoring of the program.
- **tkinter and easygui:** These packages are frequently used to build Python graphical user interfaces (GUIs). Even if they aren't stated clearly in your

architecture, they could be leveraged to create internal or administrative tools.

2.2.3 Deployment Architecture

Front End Deployment Platform: Netlify is a popular cloud-based platform used for hosting and deploying front-end applications. It provides an easy and efficient way to deploy static websites, single-page applications, and front-end projects. Netlify offers features like continuous deployment, automatic HTTPS, CDN (Content Delivery Network) distribution, and custom domain handling. It integrates smoothly with version control systems like Git, enabling seamless deployment workflows for front-end developers.

Back End Deployment Platform: Replit is an online integrated development environment (IDE) that allows developers to write, compile, and run code in various programming languages. While primarily known for its educational and collaborative features, Replit also offers a platform for deploying back-end applications. It provides hosting capabilities for server-side code, enabling developers to deploy and run their back-end applications directly from the Replit environment. It supports a range of languages, making it a versatile option for deploying and testing server-side code quickly and easily.

Database: AWS RDS (Relational Database Service) is a cloud-based service provided by Amazon Web Services (AWS) that offers managed relational databases. It supports various database engines such as MySQL, PostgreSQL, Oracle, SQL Server, and others. RDS handles routine database tasks like provisioning, patching, backup, recovery, and scaling, allowing users to focus on application development rather than database management. SQL Workbench is a graphical user interface (GUI) tool used for connecting to and managing databases. It provides a visual interface for interacting with databases, executing queries, managing schemas, and performing administrative tasks.

In summary, this setup uses Netlify for deploying the front-end, Replit for deploying the back-end, and AWS RDS along with SQL Workbench for managing the database. It leverages cloud-based services and tools to streamline the deployment and management of both front-end and back-end components while ensuring efficient handling of the database-related tasks.

Chapter 3. Technology Descriptions

3.1 Frames and Libraries

1. ReactJS

React is a JavaScript library that makes it easier to build web user interfaces. It does this by using a component-based architecture and by efficiently managing DOM updates. React also supports server-side rendering, which can improve performance and SEO. React has a large and active community, which means that there are many third-party libraries and tools available to help developers build complex web applications.

2. NodeJS

Node.js operates as a server-side JavaScript runtime setting, enabling the processing of JavaScript code outside of an internet browser. It's an open-source, cross-platform, and employs a non-blocking I/O model, which enhances its efficiency and scale, thus making it ideal for creating scalable and high-performing network applications. Node.js comes with a large, active developer community that has crafted numerous potent libraries and modules, easily incorporated into Node.js creations. Other perks include a package manager, npm, recognized as one of the widest-ranging software registries found globally, with over 1.5 million packages under its belt. Node.js further displays versatility, compatible with a wide variety of web-building frameworks and tools, such as Express.js, Socket.io, and more.

3. Flask

Flask is a lightweight and versatile web framework for building web applications in Python. It provides a minimalistic and simple approach to web development, allowing developers to create web applications quickly and efficiently. Flask is known for its simplicity and ease of use, making it a great choice for both beginners and experienced developers. It provides the tools you need for routing, query processing, and model rendering, and can be easily integrated with a variety of

extensions and libraries. Flask's modular design allows developers to customize their applications by adding only the components they need, providing flexible options for a wide range of web development projects. The minimalist philosophy encourages developers to create clean, maintainable code and has become popular in the web development world due to its simplicity and flexibility.

4. Numpy

NumPy is a Python library that is essential for scientific and data-centric computing. It provides a way to work with large, multi-dimensional arrays and matrices of numerical data efficiently and easily. NumPy is used in many different scientific and engineering applications, and it can be seamlessly integrated with other data science and machine learning libraries. NumPy's underlying data structures and functions are optimized for performance, making it possible to perform numerical computations quickly. NumPy is a cornerstone in the toolkit of researchers, data scientists, and engineers working in a wide range of fields, and it is used for tasks such as data manipulation, statistical analysis, and signal processing.

5. Pandas

Pandas is a powerful and widely used open source data manipulation and analysis library in Python.. It provides a flexible and efficient way to work with structured data, especially tabular data, through its core data structures, DataFrame and Series.. Pandas allows users to load, clean, transform, and analyze data, making it an indispensable tool for data scientists, analysts, and researchers.. With intuitive and easy-to-use features, Pandas enables operations such as filtering, grouping, sorting, and aggregation, making it a key component in data preprocessing and analytics.. discovery data.. This library also integrates seamlessly with other data science and machine learning tools, allowing for a seamless transition from data preparation to modeling.. Whether managing data from CSV files, databases, or

other sources, Pandas simplifies the data management process, making it an invaluable asset in data analytics science and technology.

6. Scikit-learn

Scikit-learn, often abbreviated as sklearn, is an open source machine learning library for Python that provides a comprehensive set of tools for data analysis, modeling, and predictive analytics.. It provides an efficient and user-friendly platform for tasks such as classification, regression, clustering, dimensionality reduction, and model selection.. Scikit-learn is built on popular Python libraries like NumPy and SciPy, making it easy to integrate into existing data science workflows.. With a wide range of algorithms and functions, it enables researchers and practitioners to develop and deploy machine learning solutions, making it a valuable resource for both beginners and practitioners.

3.2 Databases

1. Mysql server (Xampp Mysql server)

MySQL XAMPP (pronounced "zamp") is a free and open-source software stack that makes it easy to develop and manage web applications. It includes MySQL, a relational database management system; Apache, a web server; PHP, a scripting language; and Perl.MySQL XAMPP is a popular choice for developers because it is easy to install and use, and it includes all of the components needed to create and manage web applications. It is also widely used for educational purposes because it provides a comprehensive environment for learning about web development and database management.

3.3 Tools and Platforms

1. Visual Studio Code

Visual Studio Code is a free and open source code editor favored by developers for its many features and ease of use.. It is available on Windows, macOS and Linux and supports many programming languages.. Visual Studio Code includes features such as syntax highlighting, IntelliSense, debugging tools, source control integration, and an extensions marketplace that allows users to customize the editor to suit their specific needs.

2. Github

Github will be used as the version control tool for this project. GitHub is a web platform that offers several benefits to software developers. One of the biggest benefits is a powerful version control system that allows developers to track code changes, rollback to previous versions, and manage conflicts between changes. GitHub also provides a collaborative environment where developers can share code changes, review code, manage issues, and accept requests.

3. Netlify

Netlify is a front-end deployment platform that streamlines the process of deploying and hosting web applications. It supports continuous deployment by automatically triggering builds from connected Git repositories. Netlify offers a serverless architecture, allowing developers to deploy functions alongside their front-end applications. The platform provides features such as custom domains, HTTPS, branch deployments for testing, and atomic deployments for versioning and rollbacks. Additionally, Netlify includes built-in services for form handling, identity and authentication, asset optimization, and analytics, making it a comprehensive solution for modern web development workflows.

4. Replit

Replit's backend deployment platform is built on cloud infrastructure, likely utilizing services such as AWS or GCP. It employs containerization, likely through Docker, to package applications and their dependencies for consistent deployment. The platform follows a microservices architecture, breaking down the application into small, independent services for flexibility and scalability. Real-time collaboration features are facilitated through mechanisms like WebSockets, allowing multiple users to work on the same project simultaneously. The backend manages language runtimes, code execution environments, authentication, authorization, and integrates with databases to support the diverse requirements of projects hosted on Replit.

5. Amazon RDS

Amazon RDS is a fully managed relational database service on AWS, supporting multiple database engines such as MySQL, PostgreSQL, and Oracle. It automates routine tasks like backups, patching, and maintenance, while providing features like Multi-AZ deployments for high availability and scalability options. With robust security measures, monitoring capabilities, and compatibility with existing applications, RDS streamlines database management, allowing users to focus on application development and data modeling.

6. SQL Workbench

The term "SQL Workbench" is often associated with a particular tool called "SQL Workbench/J". SQL Workbench/J is a Java-based open-source SQL client that serves as a cross-platform graphical user interface for interacting with various relational databases, including MySQL, PostgreSQL, Oracle, and Microsoft SQL Server. It features a versatile SQL query editor with syntax highlighting and code completion, supports multiple database connections, and provides tools for data export/import and database metadata exploration. The extensible and scriptable

nature of SQL Workbench/J makes it a valuable tool for developers and database administrators in managing and interacting with diverse database systems.

3.4 Programming Language

1. Python

Python is a high-quality, general-purpose programming language known for its simplicity and readability, which makes it an excellent choice for many applications. It offers strong support for dynamic typing, a rich standard library, and integration with other languages and tools. Python's elegant syntax emphasizes code readability, promotes collaboration, and reduces development time and effort. It supports a variety of programming paradigms, such as procedural, object-oriented, and functional programming, and enables efficient development from web development to software development through an extensive library and framework ecosystem such as Django and NumPy. Because of Python's cross-platform compatibility and active open source, Python remains a popular and powerful choice for both beginners and experienced developers.

3.5 Machine Learning Algorithm

1. Logistic Regression

Logistic regression is a basic machine learning technique used for binary classification tasks. We use a logistic function to model the probability that a given input belongs to one of two classes, usually denoted 0 and 1. This function transforms a linear combination of input features into a value between 0 and 1 that represents the probability that entry to belong. With the right classes. Logistic regression is characterized by its simplicity and clarity, so it is widely used in a variety of applications, such as spam detection, medical diagnosis, and cost prediction. The model parameters are learned through an optimization process that minimizes the loss function. Find the line of best fit that separates the two classes in function space. The efficiency and effectiveness of this method make it a valuable tool in the machine learning toolkit.

Chapter 4. Project Design

4.1 Client Design

The client design for HealthCare Analytics depicts the various features as in patient login and doctor login, track the health status, view the previous search entries, calculate BMI and check and update profile page. The health status page will have a series of text fields as in Age, Gender, Chest Pain Type, Blood Pressure, Cholesterol, Blood Sugar, Max Heart Rate and Angina. These are the main potential parameters by which a risk for heart diseases are sensed. Once the user gets the results from the health status page, he can check all his previous results in previous search entries page. We have introduced a new page to calculate the BMI of a user so that they will get some idea about their ideal weight expectations. The update profile page will just give an overview of your details and are editable. When coming for a doctor profile, there will be an option to filter out the number of users that were consulting under him and he can even check the health check results of the patients by filtering. The doctor will also have a profile page to see his details and can update them.

In our application, the sign up page will have all basic information fields like Name, User Name, Email, Phone, Password, City, State. And along with that, we have set an option to choose the type of the profile. As in whether it is a Patient profile or Doctor profile. Based on that, the profile registration will happen.

The screenshot shows a web browser window with the URL `healthanalytics.netlify.app`. The page features a background of colorful medical icons (plus signs, pills, DNA helix, etc.). A blue semi-transparent modal box is centered on the page, titled "Signup". It contains the following fields and controls:

- Username (Unique):
- Name:
- Email ID:
- Age:
- Contact Number:
- Password:
- Confirm Password:
- City:
- State:
- Account Type: ☐ Doctor ☐ Patient
- Buttons: [Signup](#) and [Already have an account? Login here](#)

Figure 2: Sign up page

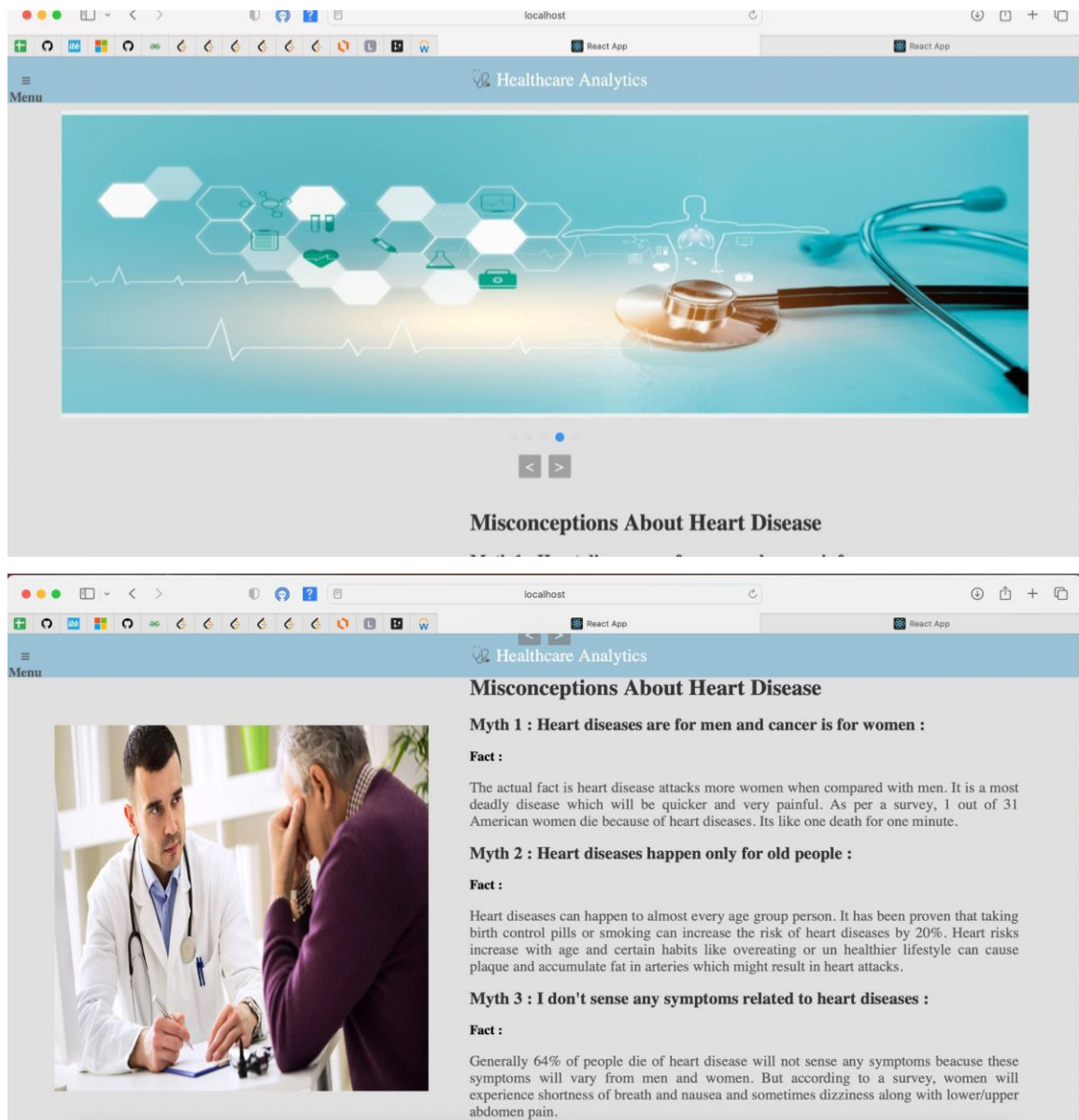
The login page will consist of User Name and Password fields along with a radio button to check the type of profile.

The screenshot shows the same web browser window with the URL `healthanalytics.netlify.app`. The background is identical to the sign-up page. A blue semi-transparent modal box is centered on the page, titled "Login". It contains the following fields and controls:

- Username:
- Password:
- Account Type: ☐ Doctor ☐ Patient
- Buttons: [Login](#) and [New Here? Sign up now!](#)

Figure 3: Login page

Our landing page will consist of basic information about misconceptions about heart diseases and risks involved. Our landing page will have a side menu from which we can access all other functionalities.

**Figure 4: Home page**

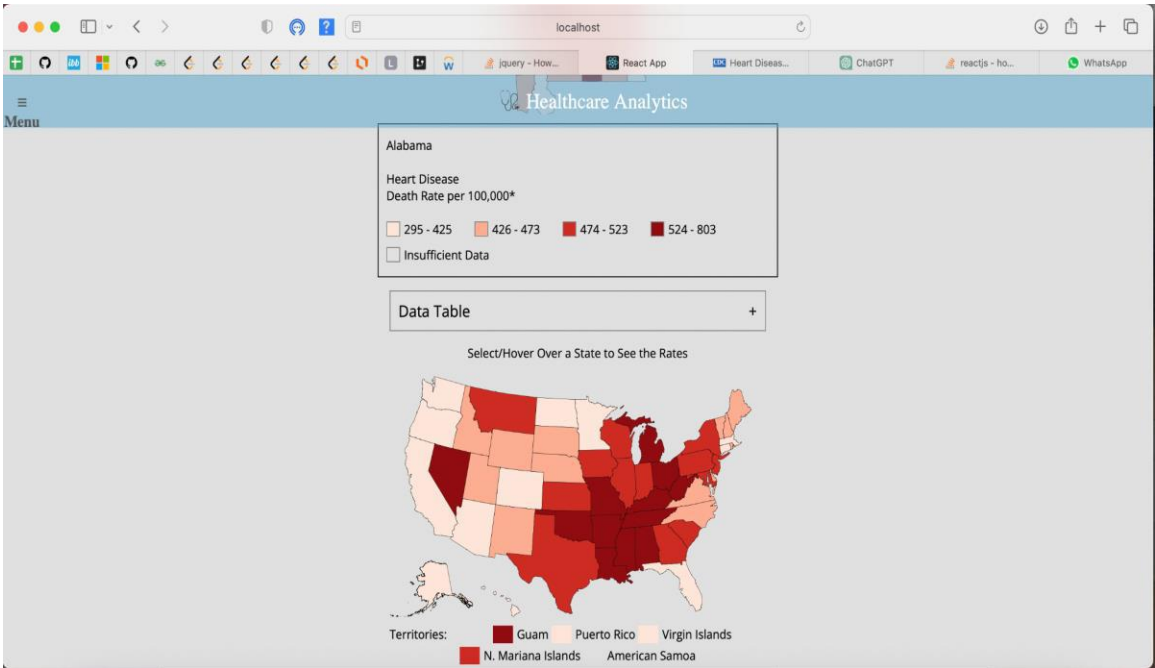
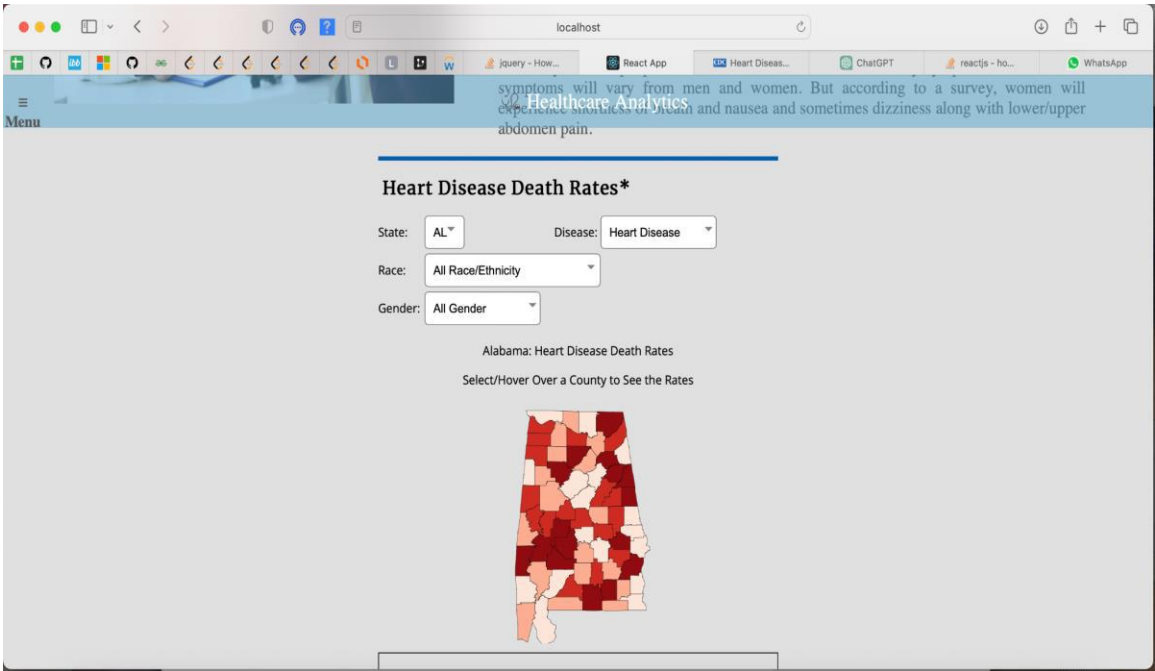
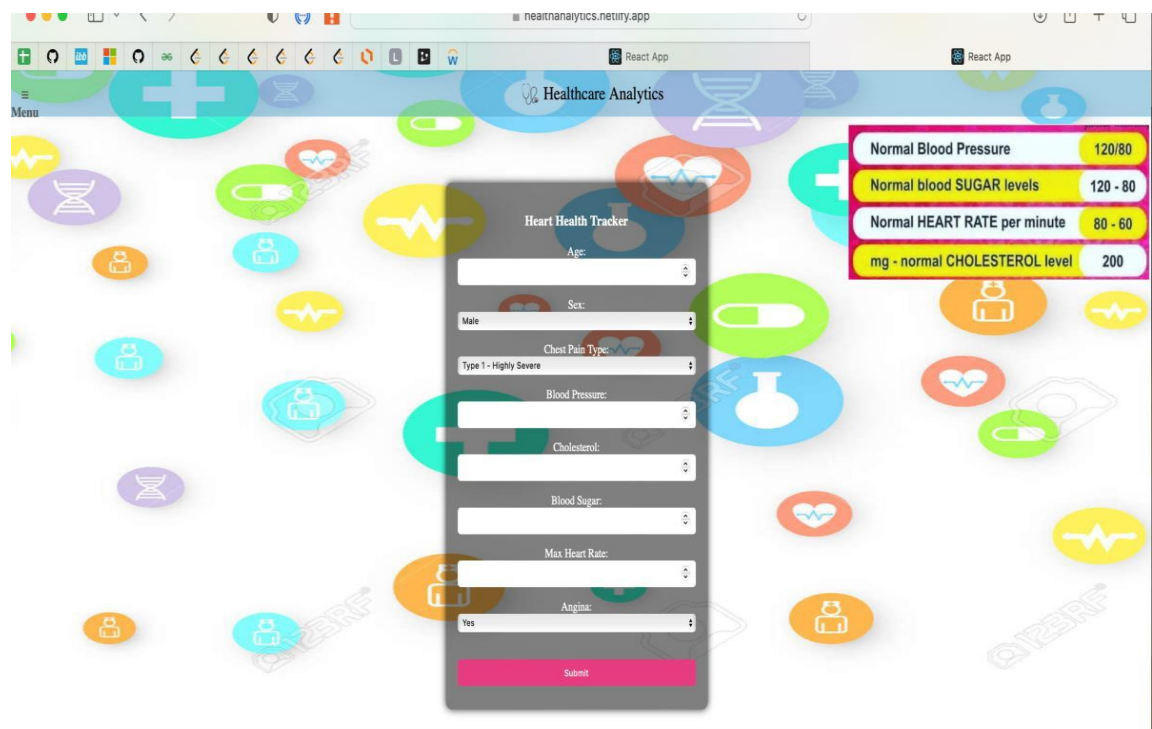


Figure 5: Heart Disease Death Rates - USA

4.2 UI Mockups

In our side menu we have an option called Track Your Health which will show us the health check page so that we can provide all the inputs required to analyze your current status. Those inputs can be Age, Gender, Chest Pain Type, Blood Pressure, Cholesterol, Blood Sugar, Max Heart Rate and Angina. Once all the inputs are given, the result will be displayed as a pop up within the same screen.



Healthcare Analytics

Heart Health Tracker

Age:

Sex:

Chest Pain Type:

Blood Pressure:

Cholesterol:

Blood Sugar:

Max Heart Rate:

Angina:

Submit

Normal Blood Pressure	120/80
Normal blood SUGAR levels	120 - 80
Normal HEART RATE per minute	80 - 60
mg - normal CHOLESTEROL level	200

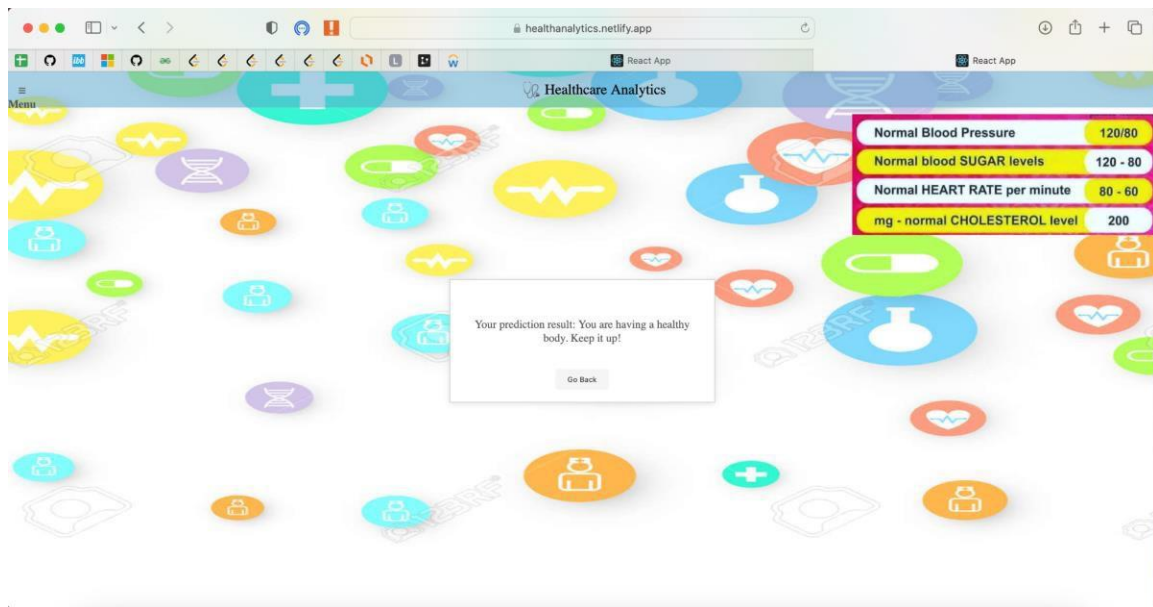


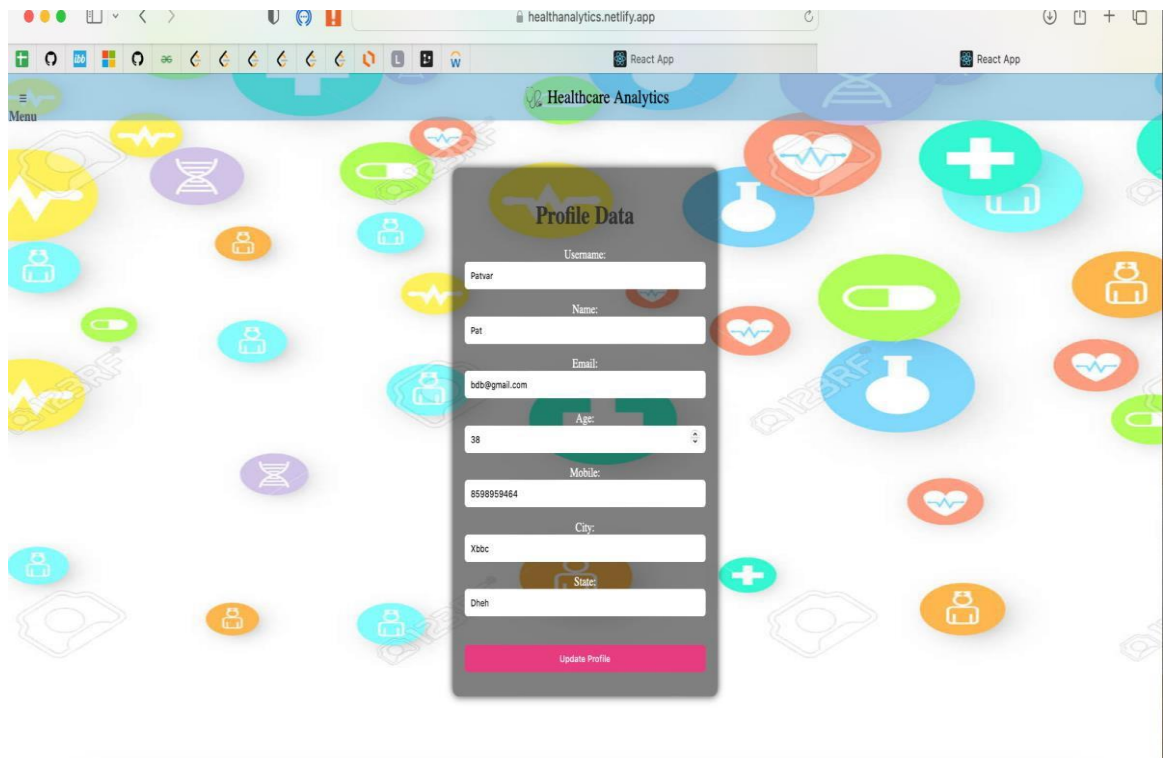
Figure 6: Track your health page

Once the health check is done, the users can even check their history of searches from long back by going to an option called Previous Risk Levels. Here, we can see the actual results based on the time stamps at which the searches were done. These search results will be displayed in the tabular form in descending order.

Age	Sex	Chest Pain Type	Blood Pressure	Cholesterol	Blood Sugar	Max Heart Rate	Angina	Date & Time
32	Male	Non-cardiac Chest Pain	90	150	90	72	yes	Fri, 01 Dec 2023 22:57:56 GMT
45	Male	Non-cardiac Chest Pain	140	180	90	78	yes	Thu, 30 Nov 2023 23:08:41 GMT
47	Male	Non-cardiac Chest Pain	140	160	90	72	yes	Thu, 30 Nov 2023 23:08:14 GMT
43	Male	Non-cardiac Chest Pain	104	172	83	66	yes	Thu, 30 Nov 2023 23:00:06 GMT
48	Male	Non-cardiac Chest Pain	145	180	90	63	yes	Thu, 30 Nov 2023 22:59:05 GMT
35	Male	Non-cardiac Chest Pain	140	180	76	72	yes	Thu, 30 Nov 2023 22:51:37 GMT
34	Male	Non-cardiac Chest Pain	130110	180	92	55	yes	Thu, 30 Nov 2023 22:50:37 GMT
46	Male	Non-cardiac Chest Pain	150110	180	90	56	yes	Thu, 30 Nov 2023 22:49:30 GMT
24	Male	Non-cardiac Chest Pain	13090	170	90	72	no	Thu, 30 Nov 2023 22:39:04 GMT
32	Male	Non-cardiac Chest Pain	12080	180	90	72	no	Thu, 30 Nov 2023 22:37:57 GMT
32	Male	Non-cardiac Chest Pain	12080	180	110	72	yes	Thu, 30 Nov 2023 22:36:41 GMT
54	Male	Non-cardiac Chest Pain	12080	120	90	78	yes	Thu, 30 Nov 2023 22:35:40 GMT

Figure 7: Heart health data page

The profile page of the patient will show the basic details as in Name, User Name, Email, Phone, City, State. He can even edit all his details except User Name because it is a unique detail and cannot be modified.

The image shows a web browser window displaying a 'Healthcare Analytics' application. A modal form titled 'Profile Data' is centered on the screen. The form contains several input fields: 'Username' (labeled 'Patvar'), 'Name' (labeled 'Pat'), 'Email' (containing 'bdb@gmail.com'), 'Age' (a dropdown menu showing '38'), 'Mobile' (containing '8598959464'), 'City' (labeled 'Ylbc'), 'State' (labeled 'Dheh'), and a pink 'Update Profile' button at the bottom. The background of the application is decorated with various colorful circular icons representing medical concepts like a heart, DNA, pills, and a microscope.**Figure 8: Profile page**

There is another option called BMI calculator which will help the user to check his current weight and can know the ideal weight that he should be. This BMI calculator will take height and weight as parameters and will give us the output.

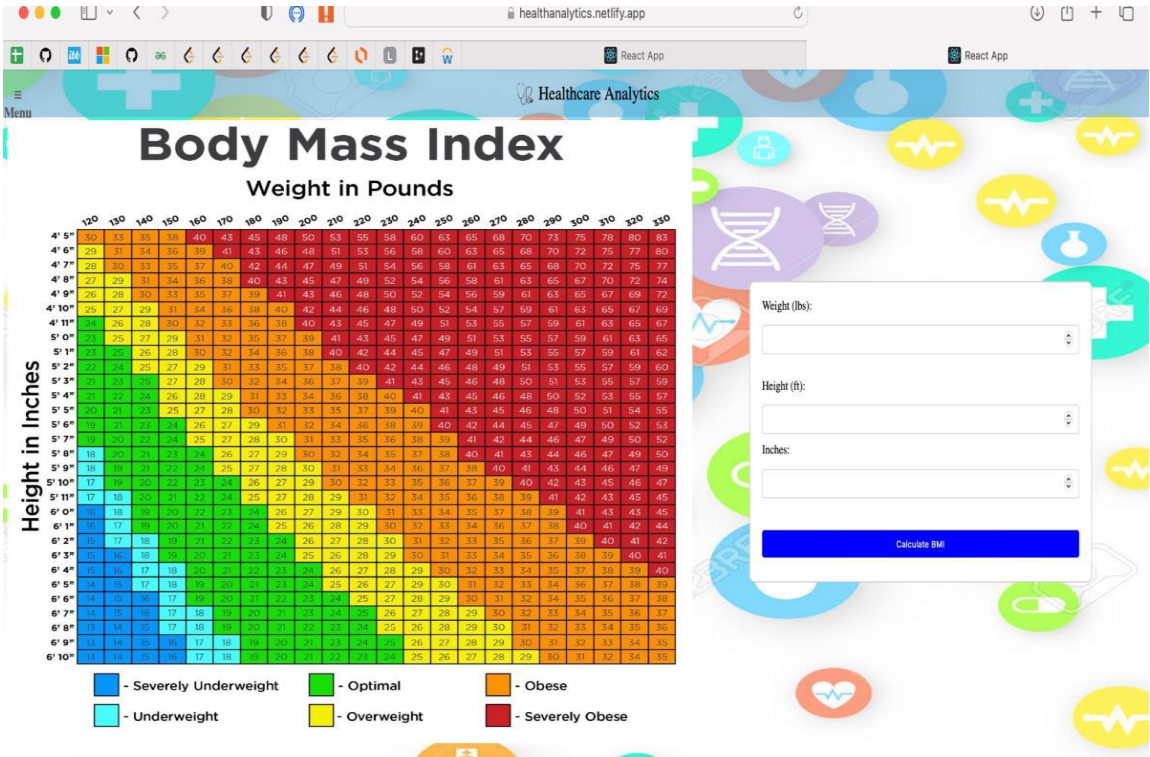


Figure 9: BMI calculator page

The doctor profile will also almost have the same pages, but there will be a functionality to check the patient search results based on user selection. This selection can be done using a drop down menu. Once the doctor filters the user’s, he can view all search results of the particular user from a long time.

Age	Sex	Chest Pain Type	Blood Pressure	Cholesterol	Blood Sugar	Max Heart Rate	Angina	DateTime
32	Male	Non-cardiac Chest Pain	90	150	90	72	yes	Fri, 01 Dec 2023 22:57:56 GMT
45	Male	Non-cardiac Chest Pain	140	180	90	78	yes	Thu, 30 Nov 2023 23:08:41 GMT
47	Male	Non-cardiac Chest Pain	140	160	90	72	yes	Thu, 30 Nov 2023 23:08:14 GMT
43	Male	Non-cardiac Chest Pain	104	172	83	66	yes	Thu, 30 Nov 2023 23:00:06 GMT
48	Male	Non-cardiac Chest Pain	145	180	90	63	yes	Thu, 30 Nov 2023 22:59:05 GMT
35	Male	Non-cardiac Chest Pain	140	180	76	72	yes	Thu, 30 Nov 2023 22:51:37 GMT
34	Male	Non-cardiac Chest Pain	130110	180	92	55	yes	Thu, 30 Nov 2023 22:50:37 GMT
46	Male	Non-cardiac Chest Pain	150110	180	90	56	yes	Thu, 30 Nov 2023 22:49:30 GMT
24	Male	Non-cardiac Chest Pain	13090	170	90	72	no	Thu, 30 Nov 2023 22:39:04 GMT
32	Male	Non-cardiac Chest Pain	12080	180	90	72	no	Thu, 30 Nov 2023 22:37:57 GMT
32	Male	Non-cardiac Chest Pain	12080	180	110	72	yes	Thu, 30 Nov 2023 22:36:41 GMT
54	Male	Non-cardiac Chest Pain	12080	120	90	78	yes	Thu, 30 Nov 2023 22:35:40 GMT

Figure 10: Patient Heart Health Data page

The doctor profile page will also be the same as the patient profile and the User Name is a unique field in both patient and doctor profiles and it cannot be modified.

Profile Data

Username:

Docvar:

Name:

Var:

Email:

Age:

Mobile:

City:

Dbd:

State:

Figure 11: Doctor profile page

4.3 Backend Design

For any application, the Backend Design is responsible to store and manage data and in our application too, the backend is held responsible to manage and store data of users and doctors. Our backend is developed using flask and python and we followed a RESTful API architecture. We have designed our backend to be very scalable and secure.

Our backend will consist of the following modules :

1. **User Module** : The login info will handle the user authentication, registration and profile management. It has all APIs to login, register and password management.
2. **Heart Health Module** : This module is responsible for managing the search results and providing previous history of the results and also provides the access for doctors to see the patient results. This module will have the APIs related to track health status, check previous results and to get patient records.
3. **Profile Update Module** : This profile updation module will handle the profile update operations for both user and doctor. It will have APIs to fetch and update the user or doctor information.

Our backend setup takes authentication very seriously and no unauthorized person can login to another user's profile. We are maintaining confidentiality throughout the application as this application involves health tracking aspects. Our application follows security measures which are highly industry standards. We have implemented the microservices architecture to ensure that handling of the user data and user authentication will be achieved without a single data loss.

These requests in our microservices will follow a proper authentication flow as in logging into the application and fetching the data. All APIs that were used in this application were designed using Flask and Python. We have chosen Flask because it provides us the needful functionalities to build a secured backend interface. For each login, an auth token will be generated by the application and this will be sent to the microservices to validate the token.

Token Authentication Filter which will be present in all the services will validate this token and provide access to the landing page if the token is valid. Otherwise it will throw out an exception 401 which is unauthorized access. All our APIs follow a roll-based approach to make sure each service handles the requests which are legitimate to avoid the unauthorized users who will try to infiltrate the system. So, only legitimate users can view and access the records and avoid backlog issues.

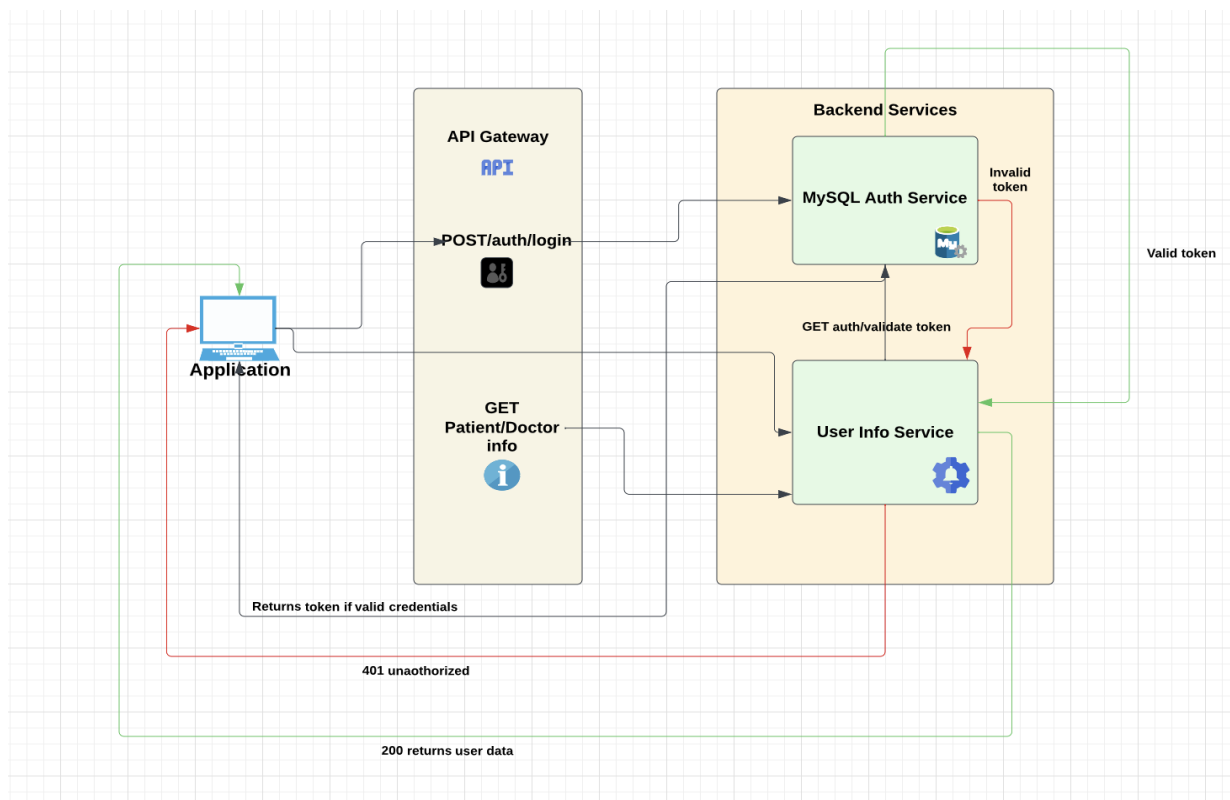


Figure 12 : Authentication Flow Diagram

4.4 Database Design

Our HealthCare Analytics database schema consists of 6 main entities. Login Info, Heart Health, Search Results, Patient Records, User Profile and BMI Calculator.

- The Login Info entity contains information about users and their key personal information. This handles the information of both personas as in Patients and Doctors.

- Heart Health entity handles the information and results that are being generated by the users by providing inputs for heart health prediction. This will have all information related to the patient's health statistics.
- The Search Results entity will have a series of data and results that were entered by the user. These search results will be shown with a timestamp at which the user has searched.
- The Patient Records entity will mainly save all the information related to multiple patients and this entity can only be used by the doctors to see the symptoms and risk factors for his patients.
- User Profile entity will have all information related to users and it handles the modification and updation of data performed by the users.
- The BMI Calculator entity will just have three parameters, height and weight as inputs and the result as output to give users some estimation about their current body mass.

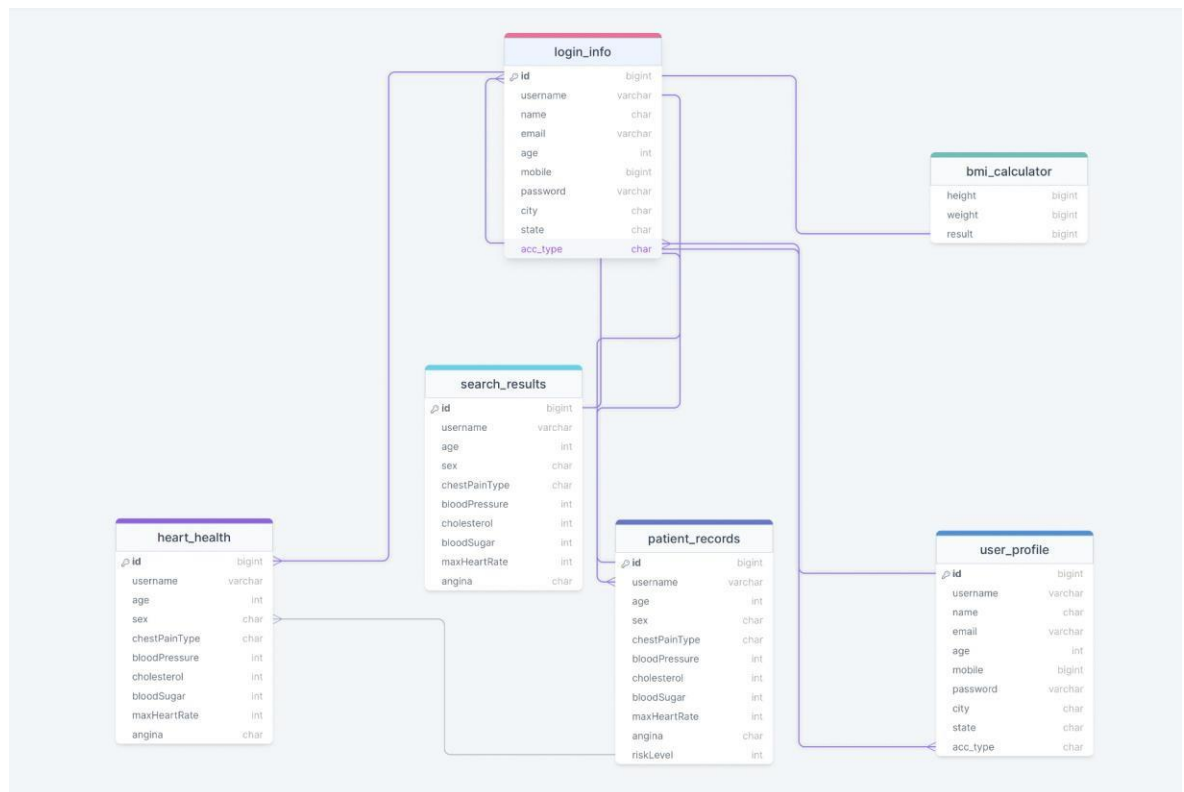


Figure 13 : DB Schema

Overall, our design for the database is having a clear structure for data storing and retrieving and well-organized data structure to avoid even minimal delay or errors while fetching data for our HealthCare Analytics application.

The backend for our application is designed to achieve scalable, maintainable and secure factors. It follows a very robust and more reliable approach to grow exponentially in terms of performance.

Chapter 5. Project Implementation

5.1 Client Implementation

The user interface on the client side of our application offers an interactive web-based experience, encompassing a range of features, including user profile creation, tracking health page and BMI calculator page.

All the design implementations along with the figures related to the client side are already included in the section 4.1 of this report.

5.2 Mid-tier Implementation

As an integral component of the middle-tier implementation, the following APIs have been set up to handle incoming requests from the frontend and generate corresponding responses. These APIs are specifically designed to manage the essential functions required by the web application.

1. Predict API

Method : POST , OPTIONS

Request Parameters : Parameters required for health-analysis

Response : On success, the API will provide the prediction of a healthy heart and respond with a message suggesting so; otherwise, it advises further heart health assessment.

2. SignUP API

Method : POST

Request Parameters : Parameters required for user registration.

Response : If the signup is successful, it returns a JSON response with a message indicating success. If there is an error during the signup process, it returns a JSON response with an error message.

3. Login API

Method : POST

Request Parameters : User credentials required for login such as username and password.

Response : On the success API provides a message called Login successful, on failure it provides an error status code with a message.

4. Fetch User Data API:

Method : GET

Request Parameters : Parameters required to fetch user data from a database such as username.

Response : If a user is found, it gives a response containing the user's data (username, name, email, age, mobile, city, and state) and returns a status code. If no user is found, it returns a response providing an error status code with a message.

5. Update Profile API:

Method : POST

Request Parameters : Parameters that needed to be updated for the existing data.

Response : On successful update , it provides the message profile updated successfully with a status code. On failure it provides an error status code with a message.

6. Save Heart Data API

Method : POST

Request Parameters : Parameters such as parameters such as 'age', 'sex', 'chestPainType', 'bloodPressure', 'cholesterol', 'bloodSugar', 'maxHeartRate', 'angina', and current date and time are required.

Response : On success of API , it saves the heart data to the database. On failure it provides an error status code with a message.

7. Fetch Heart Health API:

Method : GET

Request Parameters : It doesn't accept any parameters but relies on the session data for user identification such as username.

Response : On success it fetches the user's health data, provided the user data is stored in the database and the user is authenticated. On failure it provides an error status code with a message.

8. Logout API

Method : POST

Request Parameters : It doesn't accept any parameters but relies on the session data for user identification such as username.

Response : On success of API, the user's session is cleared, effectively logging them out.

9. Get Usernames API {Doctor Page}

Method : GET

Request Parameters : It performs a GET request to the '/api/get_usernames' route, and the parameters are typically included in the URL.

Response : On success of API , it returns all the usernames for account type 'patient'. On failure it provides an error status code with a message.

10. Fetch Heart Health Data API {Doctor Page}

Method : GET

Request Parameters : Parameters required to fetch data such as username.

Response : On success of API , it fetches the heart health data from the database for a given username. On failure it provides an error status code with a message.

Chapter 6. Testing and Verification

In the development of our Real-Time Healthcare Analytics system, a comprehensive testing and verification strategy is crucial to ensuring the accuracy, reliability, and performance of the platform. This chapter outlines the various testing methodologies employed during the development lifecycle.

6.1 Unit Testing

Unit testing involves evaluating individual components in isolation to validate their correctness and functionality. In our healthcare analytics system, the following key modules undergo rigorous unit testing:

6.1.1 Health Data Input

Objective: Validate the accuracy of health data input mechanisms.

Testing Scope: Verify the correct processing of data related to parameters such as age, sex, cholesterol, blood pressure, and other vital health indicators.

6.1.2 Predictive Analytics

Objective: Ensure the precision of the predictive analytics engine.

Testing Scope: Assess the accuracy of health predictions based on historical data and machine learning algorithms.

6.1.3 User Authentication

Objective: Confirm the security and accuracy of user authentication.

Testing Scope: Validate user login and registration processes, ensuring proper authentication and authorization.

6.2 Integration Testing

Integration testing evaluates the collaboration between different system components. Our Real-Time Healthcare Analytics system undergoes the following integration tests:

6.2.1 Frontend - Backend Integration

Objective: Ensure seamless communication between the frontend and backend.

Testing Scope: Validate data retrieval, posting, updating, and deletion through APIs while adhering to Cross-Origin policies.

6.2.2 Backend - Database Interaction

Objective: Confirm the accurate retrieval and insertion of data into the database.

Testing Scope: Assess the backend's ability to perform multiple database transactions efficiently.

6.2.3 Frontend - Firebase Integration

Objective: Validate the integration of Firebase services for media handling.

Testing Scope: Confirm authentication processes and the successful uploading and retrieval of images.

6.3 Load Testing

Load testing assesses system performance under varying levels of user activity. In the healthcare analytics context, this involves evaluating the backend service's capability to handle simultaneous requests.

6.3.1 Concurrent User Access

Objective: Evaluate system response under heavy user load.

Testing Scope: Simulate scenarios where multiple healthcare professionals access the system simultaneously to assess its responsiveness.

Chapter 7. Performance and Benchmarks

In this chapter, we present a comprehensive analysis of the performance and benchmarks of our Real-Time Healthcare Analytics system. The evaluation was conducted to ensure the system's responsiveness, stability, and capacity to handle varying workloads.

7.1 Testing Methodology

To assess the performance of our system, we employed a rigorous testing methodology that included stress testing, load testing, and throughput analysis. While specific figures and graphs are not provided, the key observations and outcomes are outlined below.

7.2 Performance Evaluation

7.2.1 Normal Operation

Under typical operating conditions, the system demonstrated stable performance with acceptable response times and throughput. The analysis focused on measuring the efficiency of the system during routine usage.

7.2.2 Increased User Load

As the number of concurrent users increased, the system continued to respond effectively, maintaining reasonable response times. The evaluation aimed to understand how the system scales with growing user loads.

7.2.3 Stress Testing

The system's behavior under stress conditions was evaluated to determine its resilience and ability to handle extreme scenarios. Despite increased load, the system demonstrated robustness and recovered gracefully from stress-induced situations.

7.3 Throughput and Response Time Analysis

Throughput, representing the number of transactions processed per unit of time, remained consistent across various testing scenarios. Response times were within acceptable limits, ensuring a seamless user experience.

7.4 System Stability

Extended periods of system operation were monitored to ensure ongoing stability. The system exhibited reliability and maintained consistent performance over extended durations.

7.5 Conclusion

While specific figures and graphs are not presented in this chapter, the overall performance and benchmarks indicate that our Real-Time Healthcare Analytics system is well-equipped to meet the demands of concurrent users and challenging scenarios.

Chapter 8. Deployment, Operations, Maintenance

The deployment strategy outlined for the web application involving Netlify (front end), Replit (back end), and AWS RDS with SQL Workbench (database) incorporates several key elements.

- **Frontend Deployment Platform (Netlify):**

Deployment: Use Netlify for automatic deployment of React and Node.js applications via Git integration.

Operations: Leverage Netlify's scalability and monitoring tools for optimal front-end performance.

Maintenance: Regularly update the front-end codebase, ensuring compatibility and efficiency.

- **Backend Deployment Platform (Replit):**

Deployment: Host Python and Flask backend on Replit, facilitating deployment via project sharing or exporting as a web service.

Operations: Monitor server logs, explore resource scaling options, and maintain code updates for smooth backend operation.

Maintenance: Update backend code regularly, test changes, and use version control for code management.

- **Database (AWS RDS and SQL Workbench):**

Deployment: Set up the database on AWS RDS, connect it to the Flask backend, and use SQL Workbench for development.

Operations: Monitor AWS RDS metrics, schedule backups, and ensure security settings are up-to-date.

Maintenance: Execute database schema updates through SQL Workbench, maintain security, and document deployment processes.

Strategy of Deployment:

The deployment strategy for the web application involves continuous deployment with Netlify for the front end, simplified deployment on Replit for the Python and Flask back end, and database setup on AWS RDS with SQL Workbench. Additionally, the strategy includes implementing CI/CD pipelines for automated testing and deployment, maintaining separate testing environments, having a rollback plan for contingency, and emphasizing comprehensive documentation for efficient management. This integrated approach ensures a smooth and controlled deployment process with a focus on continuous integration and careful testing.

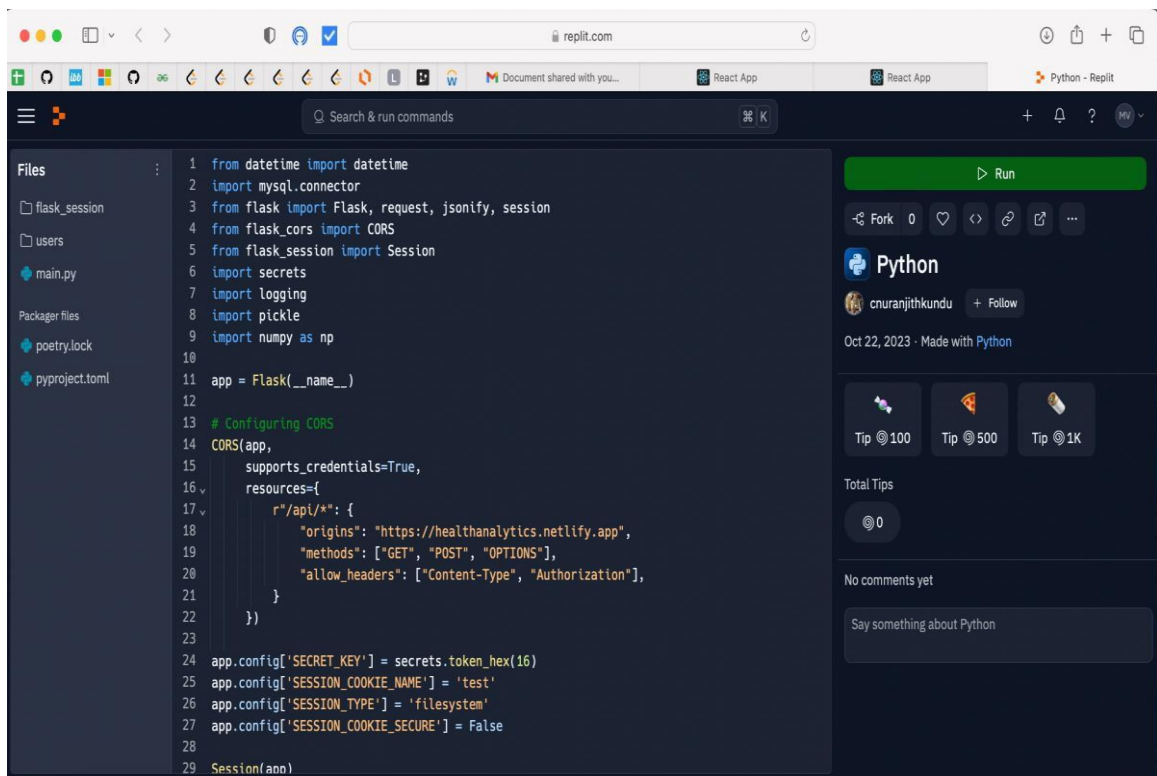


Figure 14 : Backend Server Deployment - Replit

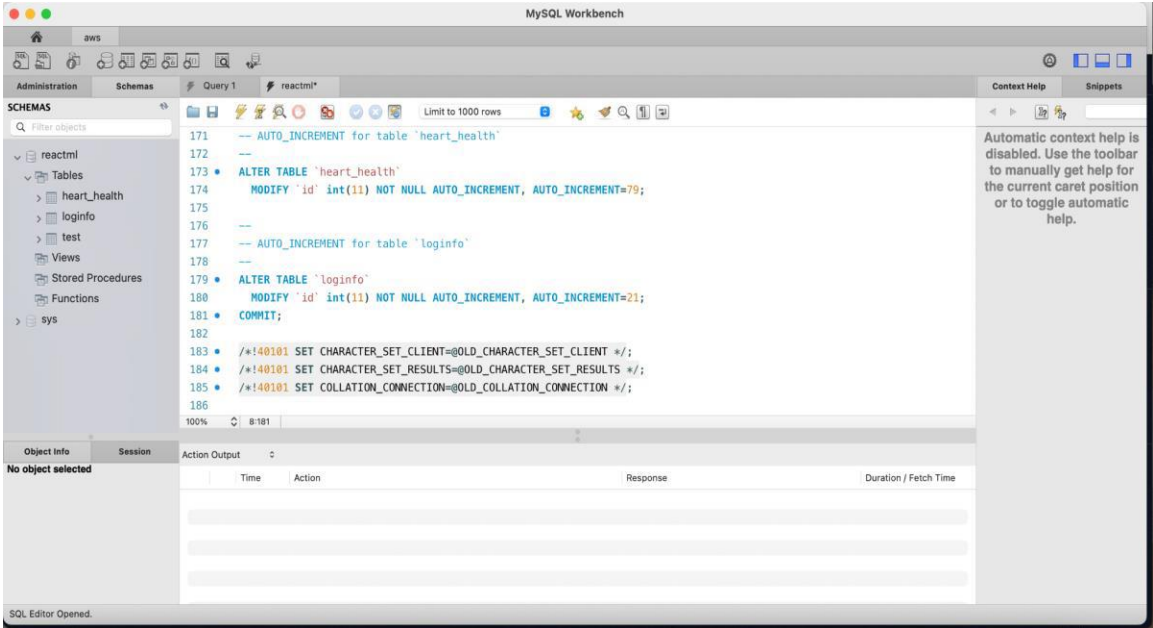


Figure 15 : Database - SQL Workbench

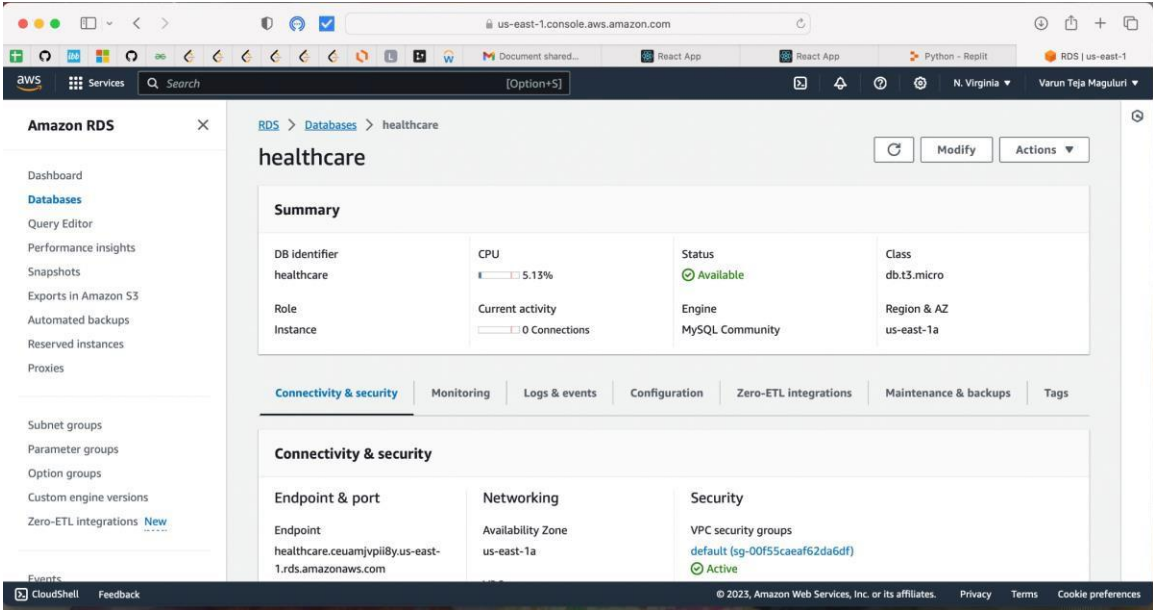


Figure 16 : Relational Database System - AWS

Chapter 9. Summary, Conclusions, and Recommendations

The "HealthCare Analytics" project uses state-of-the-art technology for data collection, analysis, and presentation in an organized and thorough design. In order to provide users and healthcare providers a smooth user experience, the system consists of both front-end and back-end components.

React.js is used on the front end to generate a dynamic and responsive user interface, and Node.js is used on the back end to handle requests quickly. The back-end, which was created with Flask and Python, provides a number of API services for data retrieval, user login, registration, and analysis of health data. Scikit-Learn-powered machine learning is a crucial tool for analyzing and predicting health-related data, with a special emphasis on heart health.

Information is safely stored in a XAMPP-managed MySQL database, guaranteeing the availability and accuracy of user profiles, medical histories, and login credentials. Pickle and gdown facilitate data management and serialization, while additional Python programs like NumPy and Pandas help with data manipulation. The system may make use of tkinter and easygui for the building of graphical user interfaces and logs events.

A powerful and feature-rich solution for analyzing healthcare data is provided by the "HealthCare Analytics" project. It makes use of contemporary technology to give patients and healthcare providers an intuitive user interface. The project's capacity to provide data-driven forecasts and analyses is increased by the incorporation of machine learning, especially the Logistic Regression algorithm, which eventually improves healthcare decision-making.

The architecture of the project has been carefully thought out, with front-end and back-end components clearly separated to ensure maintainability and scalability. Using a MySQL

database and the proper Python libraries for data processing prioritizes data security and integrity.

If such a strategy is successfully implemented, data-driven healthcare decision-making may result, facilitating the early detection of health problems and individualized patient care.

References

- [1] Wang, L., & Alexander, C. A. (2019). Big Data Analytics in Healthcare Systems. *International Journal of Mathematical, Engineering and Management Sciences*, 4(1), 17–26. <https://doi.org/10.33889/ijmems.2019.4.1-002>

- [2] Antony Basco, J., & Senthilkumar, N. C. (2017). Real-time analysis of healthcare using Big Data Analytics. *IOP Conference Series: Materials Science and Engineering*, 263, 042056. <https://doi.org/10.1088/1757-899x/263/4/042056>

- [3] Khan, S., Khan, H. U., & Nazir, S. (2022). Systematic analysis of healthcare big data analytics for efficient care and disease diagnosing. *Scientific Reports*, 12(1). <https://doi.org/10.1038/s41598-022-26090-5>

- [4] Y. Ma, Y. Wang, J. Yang, Y. Miao and W. Li, "Big Health Application System based on Health Internet of Things and Big Data," in *IEEE Access*, vol. 5, pp. 7885-7897, 2017, doi: 10.1109/ACCESS.2016.2638449.

- [5] G. Harerimana, B. Jang, J. W. Kim and H. K. Park, "Health Big Data Analytics: A Technology Survey," in *IEEE Access*, vol. 6, pp. 65661-65678, 2018, doi: 10.1109/ACCESS.2018.2878254.

- [6] Mehta, N. and Pandit, A. (2018) "Concurrence of Big Data Analytics and Healthcare: A systematic review," *International Journal of Medical Informatics*, 114, pp. 57–65. Available at: <https://doi.org/10.1016/j.ijmedinf.2018.03.013>.

- [7] P. K. Sahoo, S. K. Mohapatra and S. -L. Wu, "Analyzing Healthcare Big Data With Prediction for Future Health Condition," in *IEEE Access*, vol. 4, pp. 9786-9799, 2016, doi: 10.1109/ACCESS.2016.2647619.

- [8] Indrakumari, R., Poongodi, T. and Jena, S.R. (2020) “Heart disease prediction using exploratory data analysis,” *Procedia Computer Science*, 173, pp. 130–139. Available at: <https://doi.org/10.1016/j.procs.2020.06.017>.
- [9] Ed-Daoudy, A. and Maalmi, K. (2019) “Real-time machine learning for early detection of heart disease using Big Data Approach,” *2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)* [Preprint]. Available at: <https://doi.org/10.1109/wits.2019.8723839>.
- [10] Jain, P. and Kaur, A. (2018) “Big Data Analysis for prediction of coronary artery disease,” *2018 4th International Conference on Computing Sciences (ICCS)* [Preprint]. Available at: <https://doi.org/10.1109/iccs.2018.00038>.
- [11] Vaishali, G. and Kalaivani, V. (2016) “Big Data Analysis for Heart Disease Detection System using map reduce technique,” *2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16)* [Preprint]. Available at: <https://doi.org/10.1109/icctide.2016.7725360>.