VISUALIZING AND PREDICTING HEART DISEASE

1.INTRODUCTION

1.1 Project Overview:

One of the worst diseases in the modern era is heart disease. According to a World Health Organization [WHO] research, heart disease is one of the most dangerous illnesses for people and has been a leading cause of death globally for the past 20 years. The main challenge for medical experts is to achieve an early diagnosis of heart disease with improved accuracy because over 12 million people die each year. It is necessary to develop an application with improved disease prediction. Delivering a user-friendly website to anticipate heart disease is the aim of this project.

1.2 Purpose:

With 17.9 million deaths per year, or 31% of all deaths worldwide, cardiovascular diseases (CVDs) are the leading cause of death worldwide. The dataset utilized in this experiment contains 14 variables that can be used to predict death from heart failure, which is a typical event brought on by CVDs. In order to demonstrate the prediction of heart failure, a web server application and a prediction model utilizing machine learning are both constructed in this project. The goal of this effective heart disease prediction project is to determine whether a patient should be diagnosed with heart disease or not.

2. LITERATURE SURVEY

2.1 Existing Problem:

P.K. Anooj et al.(2011) [1] suggested a weighted fuzzy rule-based clinical decision support system (CDSS) for the diagnosis of heart disease, automatically obtaining knowledge from the patient's clinical data. It has two phases: (1) automated approach for the generation of weighted fuzzy rules and (2) developing a fuzzy rule-based decision support system. The first phase uses the mining technique, attribute selection and attribute weightage method to obtain the weighted fuzzy rules. Then, the fuzzy system is constructed in accordance with the weighted fuzzy rules and chosen attributes. Finally, the experimentation is carried out on the proposed system using the datasets obtained from the UCI repository and the performance of the system is compared with the neural network-based system utilizing accuracy, sensitivity and specificity.

Aditi Gavhane et al. (2018) [2] suggested a Neural Network model to predict heart diseases. It takes age, sex, blood pressure, heart rate, diabetes, cholestral and BMI as input into the Multilayer Perceptron algorithm. The sensors like AliveKor, MyHeart, HealthGear and Fitbit generate the parameters for the algorithm.

K.Mathan et al.(2018) [3] suggested a decision tree data mining approach with a Neural network classifier for the prediction of heart disease.Among the various prediction models the Neural networks and Gini index prediction model results in accurate prediction.A multi-layer perceptron neural networks (MLPNN) is utilized.The calculation depends on the decision trees.The most noteworthy precision accomplished is 86.1% by the equivalent width Gain ratio decision tree.

Ashok Kumar Dwivedi et al.(2018) [4] performance evaluation of different machine learning techniques for prediction of heart disease, six machine learning techniques have been applied including artificial neural network (ANN), support vector machine (SVM), logistic regression, k-nearest neighbor (KNN), classification tree and Naive Bayes. Moreover, the performance was compared using receiver operating characteristic (ROC) and calibration graph, the highest classification accuracy of 85% was reported using logistic regression with sensitivity and specificity of 89 and 81%.

C. Beulah Christalin Latha et al.(2019) [5] improving the accuracy of prediction of heart disease risk based on ensemble classification techniques, to improve the performance, weak classifiers and ensemble algorithms are used, this work has used ensemble algorithms such as bagging, boosting, voting, and stacking. Some of the techniques used for such prediction problems are the Support Vector Machines (SVM), Neural Networks, Decision Trees, Regression and Naive Bayes classifiers. This work has used ensemble algorithms such as bagging,boosting, voting, and stacking.

Davide Chicco et al. (2020) [6] used Gene Expression Omnibus dataset to predict heart disease. Random Forest Classifier with enhanced feature elimination method is used to identify the genes involved in heart failure. This system works well with an imbalanced dataset. Matthews Correlation Coefficient (MCC) and area under the receiver operating characteristic curve (ROC AUC) is used to evaluate the efficiency of the classifier model.

R. Valarmathi et al. (2021) [7] proposed a prediction system to detect heart disease which involves hyper parameter tuning of Random Forest Classifier and XGBoost Classifier model. Cleveland Heart Disease dataset (CHD) and Z-Alizadeh Sani dataset is used for the evaluation process. The performance of the algorithm is analyzed using Bayesian Optimization based on the Gaussian process. The parameters are tuned using methods like Grid Search, Randomized Search and Tpot Classifier. The random forest model with TPOT classifier gives the highest accuracy of 97.52% for the CHD dataset.

Awais Mehmood et al.(2021) [8] propose a method named CardioHelp which predicts the probability of the presence of cardiovascular disease in a patient by incorporating a deep learning algorithm called convolutional neural networks (CNN). The proposed method is concerned with temporal data modeling by utilizing CNN for HF prediction at its earliest stage.The heart disease dataset is compared with the results of state-of-the-art methods and achieved good results. Experimental results show that the proposed method outperforms the existing methods in terms of performance evaluation metrics. The achieved accuracy of the proposed method is 97% for the CHD dataset.

Deepika D et al.(2021) [9] suggested an optimized unsupervised technique for feature selection and novel Multi-Layer Perceptron for Enhanced Brownian Motion based on Dragonfly Algorithm (MLP-EBMDA) for classification of heart disease.approach.Classification has been performed by multi-layer perceptron incorporated with enhanced Brownian motion on the basis of dragonfly algorithm.The analytical results explored that the proposed system has shown effective results than the traditional methods in terms of accuracy for predicting the heart disease. The proposed system revealed prediction accuracy at the rate of 94.28% and sensitivity as 98.92%, thus resulting in better prediction of heart disease as normal or abnormal.

Md Mamun Ali et al.(2021) [10] found that using a heart disease dataset collected from Kaggle three-classification based on k-nearest neighbor (KNN), decision tree (DT) and random forests (RF) algorithms the RF method achieved 100% accuracy along with 100% sensitivity and specificity.Thus the significant rate of incorrectly diagnosed cases which could be addressed by developing accurate and efficient early-stage heart disease prediction by analytical support of clinical decision-making with digital patient records.

2.2 References:

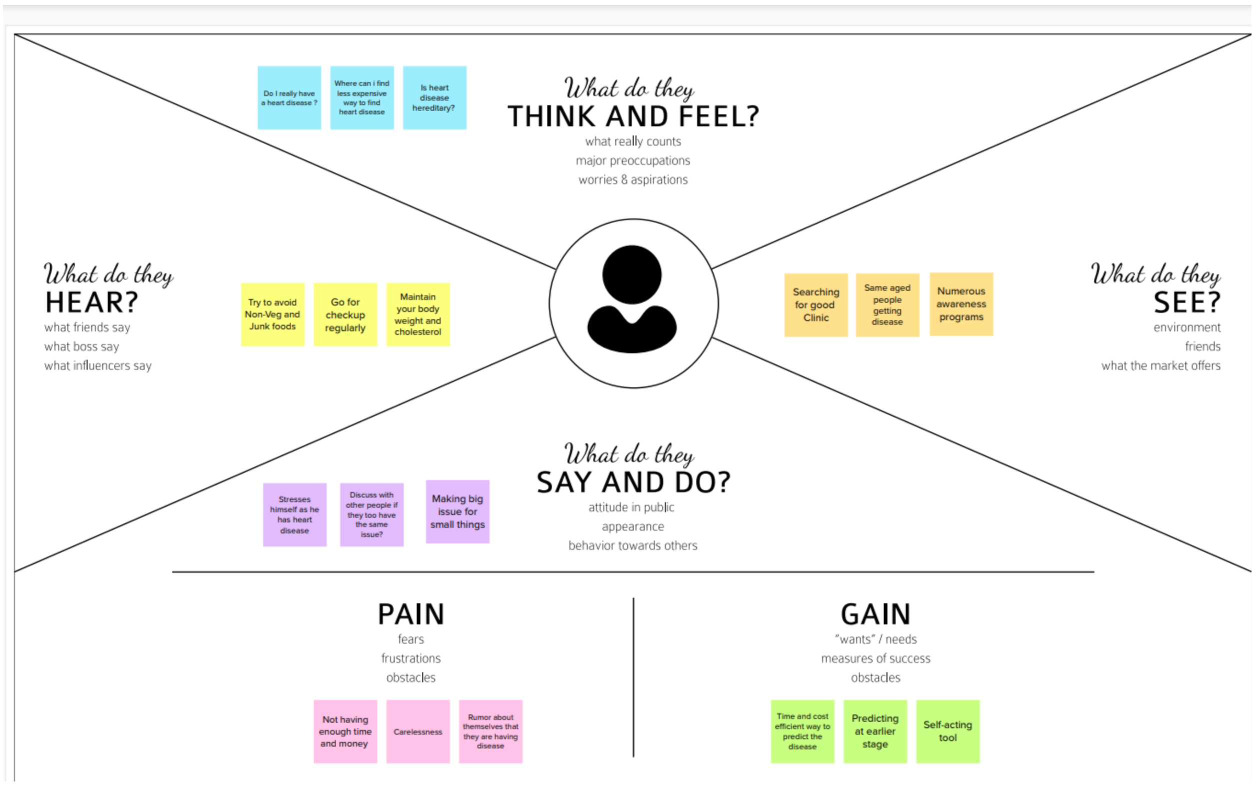
1. P K Anooj, “Clinical decision support system: Risk level prediction of heart disease using weighted fuzzy rules”, 2011, Journal King Saud University [1].
2. Aditi Gavhane, Gouthami Kokkula, Isha Pandya, Prof. Kailas Devadkar (PhD), “Prediction of Heart Disease Using Machine Learning”, 2018, IEEE Conference [2].
3. K Mathan, Priyan Malarvizhi Kumar, Parthasarathy Panchatcharam, Gunasekaran Manogaran, R. Varadharajan, “A novel Gini index decision tree data mining method with neural network classifiers for prediction of heart disease”, 2018, Springer [3].
4. Ashok Kumar Dwivedi, “Performance evaluation of different machine learning techniques for prediction of heart disease”, 2018, Neural Comput & Applic [5].
5. C. Beulah Christalin Latha, S. Carolin Jeeva, “Improving the accuracy of prediction of heart disease risk based on ensemble classification techniques, 2019, Elsevier [5].
6. Davide Chicco and Luca Oneto, “An enhanced Random Forests approach to predict heart failure from small imbalanced gene expression data, 2020, IEEE.
7. R Valarmathi, T Sheela, “Heart disease prediction using hyper parameter optimization (HPO) tuning, 2021, Elsevier [7].
8. Awais Mehmood, Munwar Iqbal, Zahid Mehmood, Aun Irtaza, Marriam Nawaz, Tahira Nazir, Momina Masood, “Prediction of Heart Disease Using Deep Convolutional Neural Networks”, 2021, Arabian Journal For Science and Engineering [8].
9. Deepika D, Balaji N, “Effective heart disease prediction using novel MLP-EBMDA approach, 2021, Elsevier [9].
10. Md Mamun Ali, Bikash Kumar Paul, Kawsar Ahmed, Francis M. Bui, Julian M, W. Quinn, Mohammad Ali Moni, “Heart disease prediction using supervised machine learning algorithms: Performance analysis and comparison”, 2021, Elsevier [10].

2.3 Problem Statement Definition:

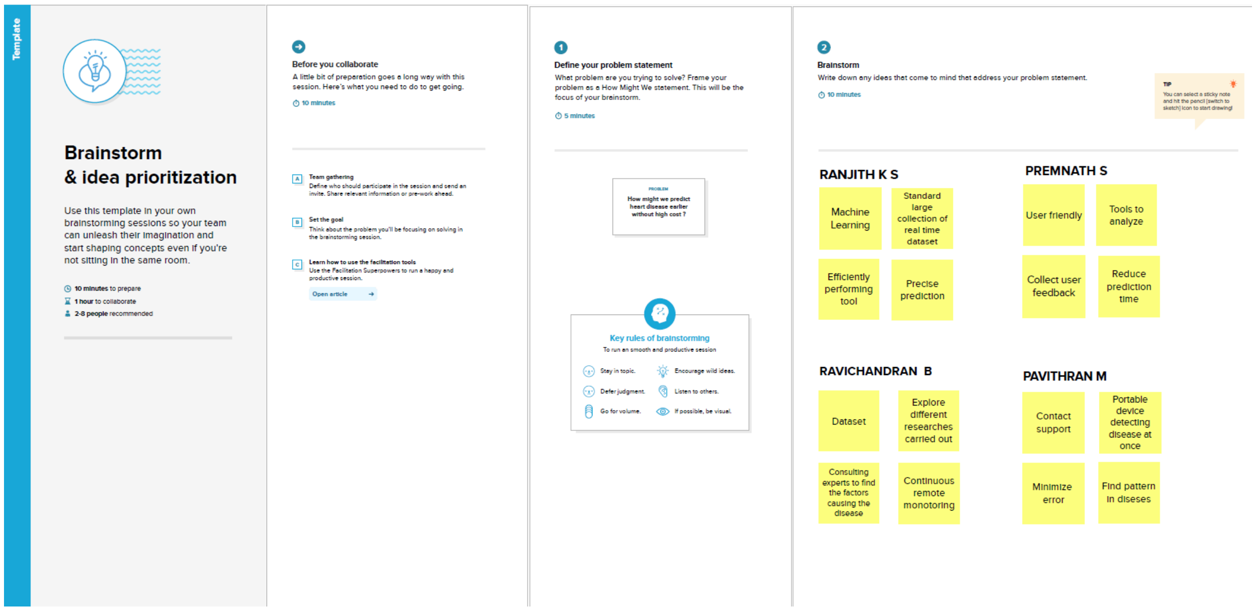
People with unhealthy lifestyles, age above 40, obesity and even the ancestors who got the disease (as the heart disease is hereditary). The issue occurs mostly for people with age above 40 and unhealthy lifestyles. The issue is originating from an unhealthy style. It mostly occurs in the blood valves of the heart. If we don't solve the problem, most of the people will die at a young age. The death rate will increase very rapidly. We should predict the problem prior to giving treatment to the patients. As the problem is predicted early, we can solve it easily.

3. IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canva:


Do I really have a heart disease ?
Where can i find less expensive way to find heart disease
Is heart disease hereditary?
What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
What do they HEAR?
what friends say
what boss say
what influencers say
Try to avoid Non-Veg and Junk foods
Go for checkup regularly
Maintain your body weight and cholesterol
Searching for good Clinic
Same aged people getting disease
Numerous awareness programs
What do they SEE?
environment
friends
what the market offers
Stresses himself as he has heart disease
Discuss with other people if they too have the same issue?
Making big issue for small things
What do they SAY AND DO?
attitude in public
appearance
behavior towards others
PAIN
fears
frustrations
obstacles
Not having enough time and money
Carelessness
Rumor about themselves that they are having disease
GAIN
"wants" / needs
measures of success
obstacles
Time and cost efficient way to predict the disease
Predicting at earlier stage
Self-acting tool

3.2 Ideation And Brainstorming:



Template

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Before you collaborate
A little bit of preparation goes a long way with this session. Here's what you need to do to get going.
10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

Open article →

1 **Define your problem statement**
What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.
5 minutes

PROBLEM
How might we predict heart disease earlier without high cost ?

Key rules of brainstorming
To run an smooth and productive session

Stay in topic. | Encourage wild ideas.
Defer judgment. | Listen to others.
Go for volume. | If possible, be visual.

2 **Brainstorm**
Write down any ideas that come to mind that address your problem statement.
10 minutes

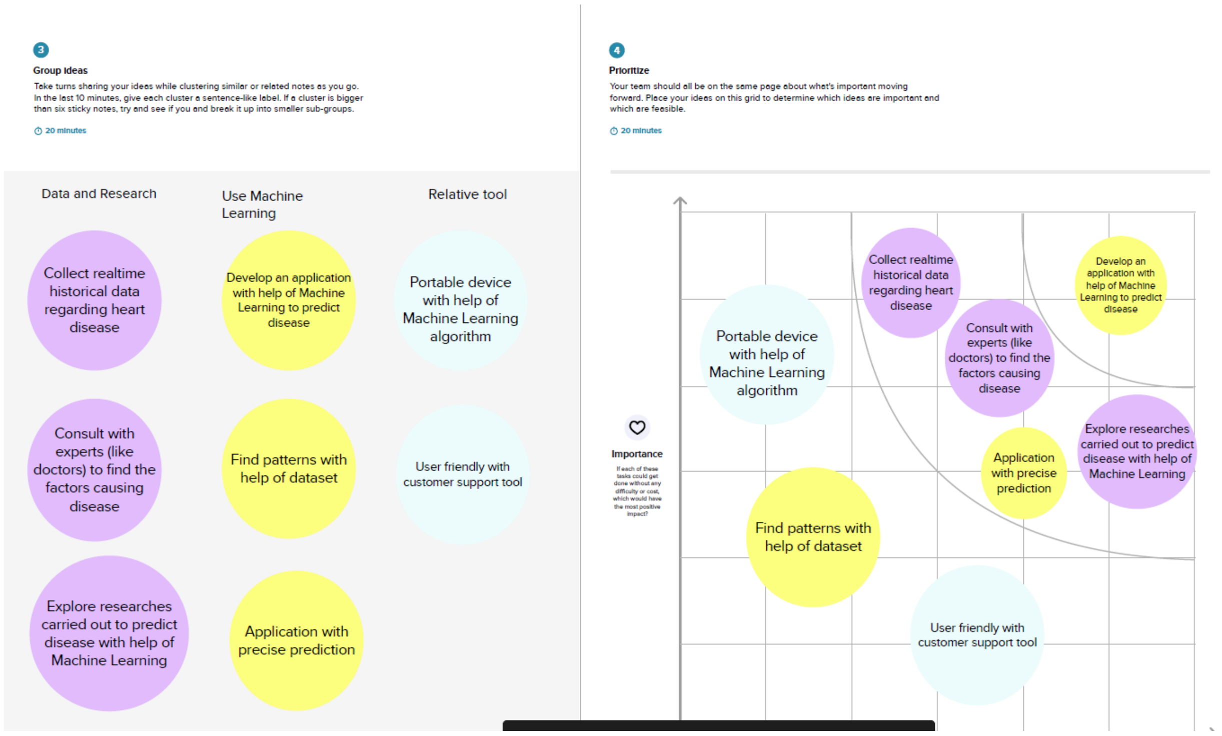
TIP
You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

RANJITH K S
Machine Learning
Standard large collection of real time dataset
Efficiently performing tool
Precise prediction

PREMNATH S
User friendly
Tools to analyze
Collect user feedback
Reduce prediction time

RAVICHANDRAN B
Dataset
Explore different researches carried out
Consulting experts to find the factors causing the disease
Continuous remote monotoring

PAVITHRAN M
Contact support
Portable device detecting disease at once
Minimize error
Find pattern in diseses



3 **Group ideas**
Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.
20 minutes

Data and Research	Use Machine Learning	Relative tool
Collect realtime historical data regarding heart disease	Develop an application with help of Machine Learning to predict disease	Portable device with help of Machine Learning algorithm
Consult with experts (like doctors) to find the factors causing disease	Find patterns with help of dataset	User friendly with customer support tool
Explore researches carried out to predict disease with help of Machine Learning	Application with precise prediction	

4 **Prioritize**
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.
20 minutes

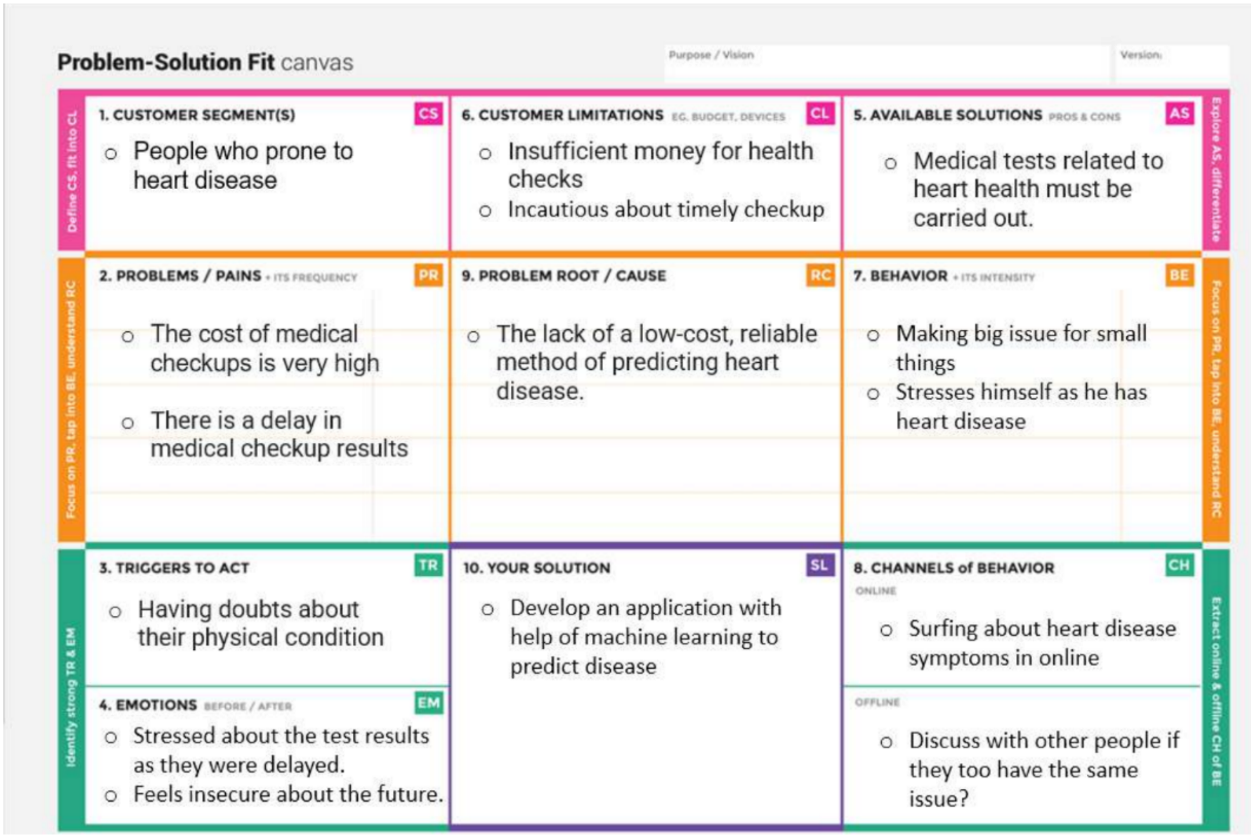
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Portable device with help of Machine Learning algorithm
Collect realtime historical data regarding heart disease
Consult with experts (like doctors) to find the factors causing disease
Develop an application with help of Machine Learning to predict disease
Explore researches carried out to predict disease with help of Machine Learning
Application with precise prediction
Find patterns with help of dataset
User friendly with customer support tool

3.3 Proposed Solution:

S.NO	Parameter	Description

1	Problem Statement (Problem to be solved)	How might we predict heart disease earlier without high cost?
2	Idea / Solution description	Develop an application with the help of Machine Learning to predict disease. Collect real time historical data regarding heart disease. Consult with experts (like doctors) to find the factors causing disease. With the above statements a machine learning algorithm is developed.
3	Novelty / Uniqueness	The solution is unique as it requires minimum effort to predict precisely.
4	Social Impact / Customer Satisfaction	The application provides results at low cost compared to now.User friendly,can predict as early as possible.

3.4 Problem Solution Fit:



Problem-Solution Fit canvas

Purpose / Vision

Version:

Define CS, fit into CL

1. CUSTOMER SEGMENT(S) CS
- People who prone to heart disease

6. CUSTOMER LIMITATIONS EG. BUDGET, DEVICES CL
- Insufficient money for health checks
- Incautious about timely checkup

5. AVAILABLE SOLUTIONS PROS & CONS AS
- Medical tests related to heart health must be carried out.

Explore AS, differentiate

Focus on PR, tap into BE, understand RC

2. PROBLEMS / PAINS + ITS FREQUENCY PR
- The cost of medical checkups is very high
- There is a delay in medical checkup results

9. PROBLEM ROOT / CAUSE RC
- The lack of a low-cost, reliable method of predicting heart disease.

7. BEHAVIOR + ITS INTENSITY BE
- Making big issue for small things
- Stresses himself as he has heart disease

Focus on PR, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TO ACT TR
- Having doubts about their physical condition

4. EMOTIONS BEFORE / AFTER EM
- Stressed about the test results as they were delayed.
- Feels insecure about the future.

10. YOUR SOLUTION SL
- Develop an application with help of machine learning to predict disease

8. CHANNELS of BEHAVIOR CH

ONLINE
- Surfing about heart disease symptoms in online

OFFLINE
- Discuss with other people if they too have the same issue?

Extract online & offline CH of BE

4. REQUIREMENT ANALYSIS

4.1 Functional Requirements:

FR No	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Email
FR-2	User Confirmation	Confirmation via Email
FR-3	Visualizing Data	Visualize the presence of heart disease through Dashboard created using IBM Cognos Analytics
FR-4	Generation Report	Users can view their reports
FR-5	Disease Prediction	Users can predict disease presence.

4.2 Non Functional Requirements:

NFR No	Non-Functional Requirement	Description
NFR-1	Usability	Provide a simplified user interface to access contents easily.
NFR-2	Security	Have a backup dataset. User reports should be accessed only by the respective users.
NFR-3	Reliability	Must work without error or minimum error
NFR-4	Performance	It is affected by the implementing algorithm. Depending on the error metrics we have to choose an algorithm with high response time.
NFR-5	Availability	Must be available for the user 24 x 7 without

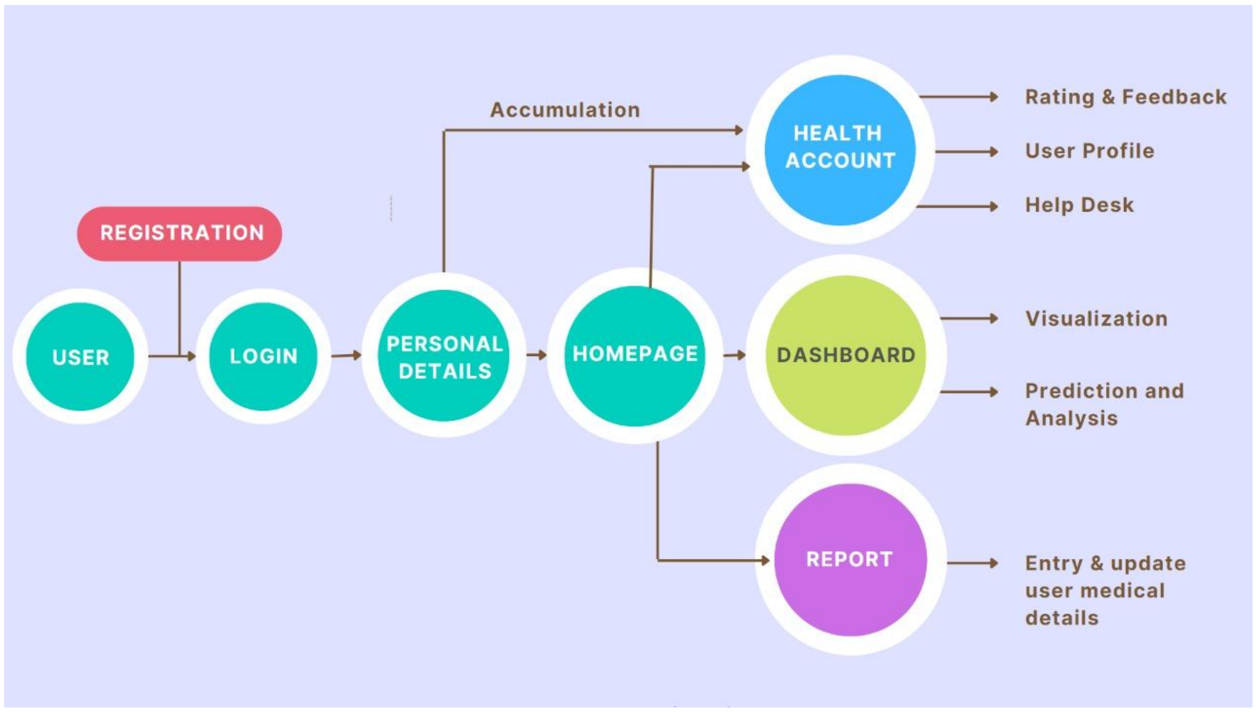
		interruptions. Must be accessible for all types of users(mobile, laptop, etc.,)
NFR-6	Scalability	Should withstand a high number of users and large datasets.

5.PROJECT DESIGN

5.1 Data Flow Diagram:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

Data Flow Diagram for Heart Disease Prediction Dashboard:



Flow:

- User creates an account in the application.
- User enters the medical records.
- Users can view the visualizations of trends in the form of graphs and charts for his/her medical records with the trained dataset.
- Users can view the probability of occurrence of heart disease in the dashboard.

5.2 Solution And Technical Architecture:

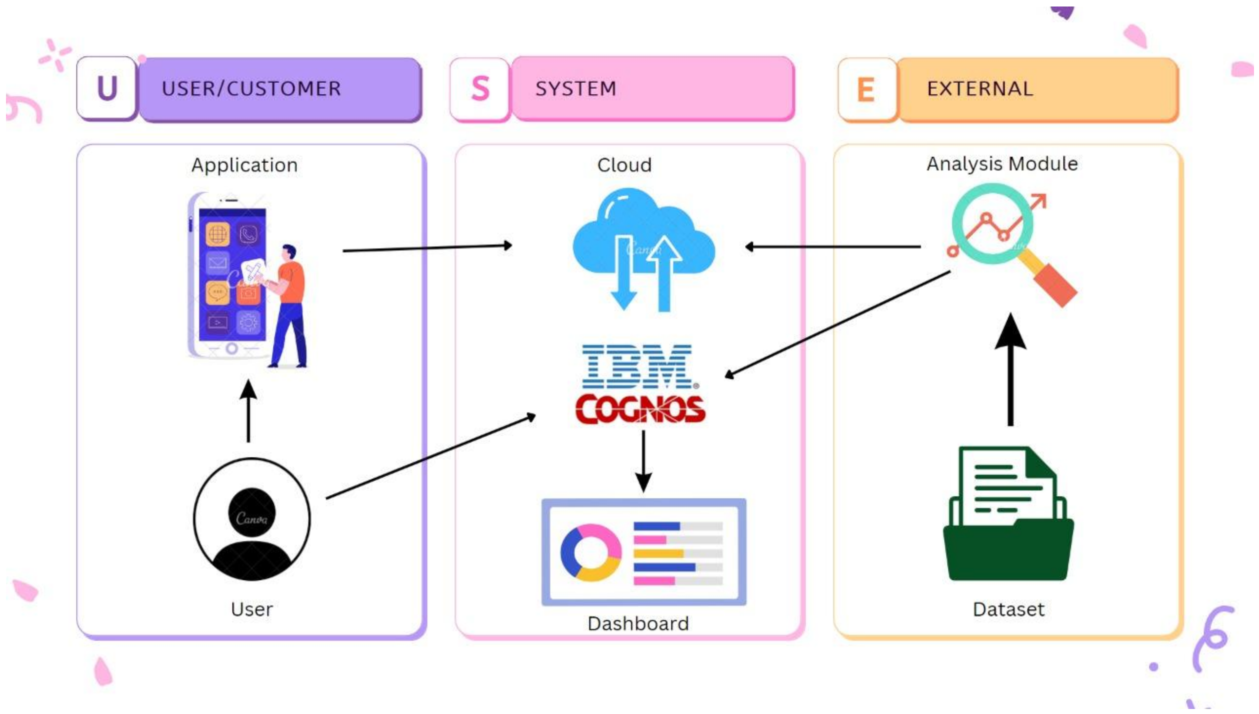


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	Importing data	Data is imported from external sources and is used for the analytics.	Python, Numpy , Pandas
2.	Data Cleaning	Data cleaning is a process by which inaccurate, poorly formatted, or otherwise messy data is organized and corrected	Python
3.	Data Pre-processing	Data pre-processing, a component of data preparation, describes any type of processing performed on raw data to prepare it for another data processing procedure	Python
4.	Training data	Training data is the subset of original data that is used to train the machine learning model	Python
5.	Testing data	Test data is data which has been used to check the accuracy of the ML model.	Python

6.	Machine Learning model	A machine learning model is an algorithm that predicts the disease from the data.	Python, Sk learn
7.	Improve model performance	Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right	Python
8.	Checking accuracy	A data accuracy check is a set of quality validations that take place before using data.	Python

Table-2: Application Characteristics:

S.No	**Characteristics**	**Description**	**Technology**
1.	Open-Source Frameworks	Frameworks are about more than just creating a development environment. They help to define a set of standards that programmers can follow when working collectively. When programmers choose a certain framework, they adopt the specific tools and methodologies associated with that framework. This also means they must be mindful of your choice, as they may end up with processes that don't fit the needs of their project or the developers involved.	ReactJs
2.	Security Implementations	IAM Controls and Encryptions are implemented to improve security of the application.	Encryptions, IAM Controls.
3.	Scalable Architecture	Scalable operations are implemented using APIs like HTTP, HTTPS.	API Gateway
4.	Availability	To ensure high availability and optimal service, the load balancer performs continual health checks of each server in the cluster, using probes to determine its eligibility for requests.	Server Load Balancers

5.	Performance	Performance of the system is increased using caching methodology.	Caching

5.3 User Stories:

User Type	Functional Requirement	User Story Number	User Story / Task	Acceptance Criteria	Priority	Release
Customer (Web User)	Registration	USN - 1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / Dashboard	High	Sprint - 1
		USN - 2	As a user, I will receive a confirmation email once I have registered for the application.	I can receive confirmation email & click confirm.	High	Sprint - 1
	Login	USN - 3	As a user, I can log into the application by entering	I can access my account / Dashboard when logged in.	High	Sprint-1

			email & password.			
Customer (Web User)	Profile	USN - 4	As a user, I can manage my user profile.	I can view, edit my personal details.	High	Sprint - 1
Customer (Web User)	Dashboard	USN - 5	As a user, I can view my complete medical analysis.	I can view my medical analysis in the dashboard	High	Sprint - 2
		USN - 6	As a user, I can view the probability of occurrence of heart disease	I can view the probability of occurrence of heart disease using a dashboard.	High	Sprint - 2
Customer	Predictor	USN - 7	As a user, I can predict my presence of heart disease	I can view the prediction result.	High	Sprint - 2
Administrator	User Profile	USN - 8	As an admin, I can manage profiles of the users.	I can view, edit and delete user accounts of the user.	High	Sprint - 3
Customer (Web User)	Queries	USN - 9	As a user, I can ask queries.	I can ask queries and get resolved by the answers given by the helpdesk.	Medium	Sprint - 3
Customer Care	Helpdesk	USN - 10	As a customer	I can view the	Medium	Sprint - 4

Executive			care executive, I can view the questions asked by the users.	questions asked by the users ordered by time of the question asked and filter the questions based on responses.		
		USN - 12	As a customer care executive, I can answer the questions asked by the users.	I can respond to the questions asked by the users.	Medium	Sprint - 4
Customer (Web User)	Rating	USN - 12	As a user, I can rate the website and provide feedback.	I can rate and provide feedback on the website.	Low	Sprint - 4

6.PROJECT DESIGN

6.1 Sprint Planning and Estimation:

Sprint	**Functional Requirement**	**User Story Number**	**User Story / Task**	**Story Points**	**Priority**	**Team Members**
Sprint - 1	Registration	USN - 1	As a user, I can register for the application by entering my email, password, and confirming my password.	5	High	Premnath S
Sprint - 1		USN - 2	As a user, I will receive a confirmation email once I have registered for the application.	5	High	Ravichandran B
Sprint - 1	Login	USN - 3	As a user, I can log into the application by entering email & password.	5	High	Pavithran M
Sprint - 1	Profile	USN - 4	As a user, I can manage my user profile.	5	High	Ranjith KS
Sprint - 2	Dashboard	USN - 5	As a user, I can view my complete	5	High	Pavithran M

			medical analysis.			
Sprint - 2		USN - 6	As a user, I can view the probability of occurrence of heart disease	5	High	Ranjith K S
Sprint - 2	Predictor	USN - 7	As a user, I can predict the present of heart disease	10	High	Ranjith K S
Sprint - 3	User Profile	USN - 8	As an admin, I can manage profiles of the users.	10	High	Ravichandran B
Sprint - 3	Queries	USN - 9	As a user, I can ask queries.	10	Medium	Premnath S
Sprint - 4	Helpdesk	USN - 10	As a customer care executive, I can view the questions asked by the users.	8	Medium	Premnath S
Sprint - 4		USN - 11	As a customer care executive, I can answer the questions asked by the users.	8	Medium	Ravichandran B
Sprint - 4	Rating	USN - 12	As a user, I can rate the website and provide feedback.	4	Low	Pavithran M

6.2 Sprint Delivery Schedule:

Sprint	**Total Story Points**	**Duration**	**Sprint Start Date**	**Sprint End Date (Planned)**	**Story Points Completed (as on Planned End Date)**	**Sprint Release Date (Actual)**
Sprint 1	20	6 days	31 Oct 2022	05 Nov 2022		
Sprint 2	20	6 days	07 Nov 2022	12 Nov 2022		
Sprint 3	20	3 days	14 Nov 2022	16 Nov 2022		
Sprint 4	20	3 days	17 Nov 2022	19 Nov 2022		

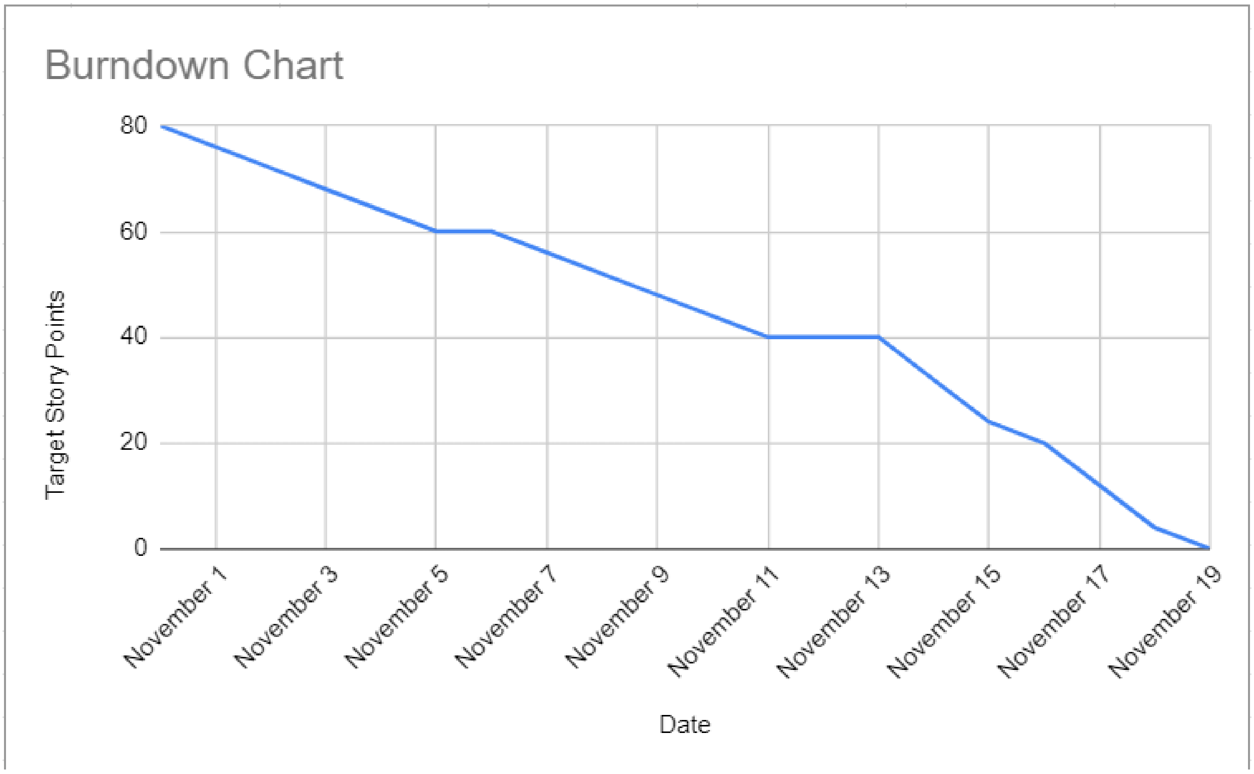
Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20(points per sprint). Let’s calculate the team’s average velocity (AV) per iteration unit (story points per day).

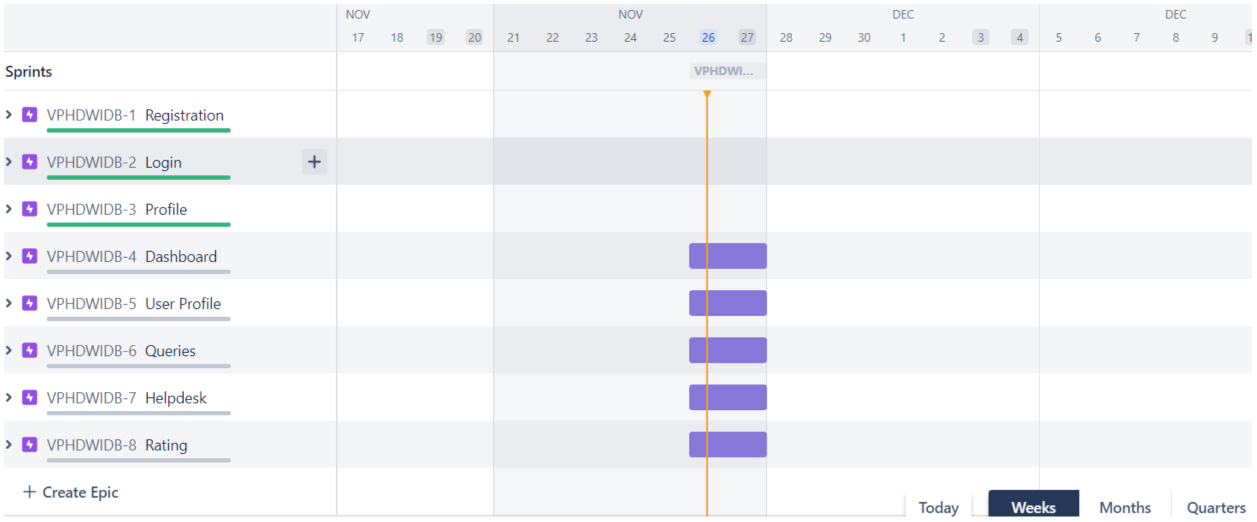
AV = sprint duration/velocity = 20/10 = 2

Burndown Chart:

A burndown chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



6.3 JIRA Reports:



VPHDWIDB-1 Registration
VPHDWIDB-2 Login
VPHDWIDB-3 Profile
VPHDWIDB-4 Dashboard
VPHDWIDB-5 User Profile
VPHDWIDB-6 Queries
VPHDWIDB-7 Helpdesk
VPHDWIDB-8 Rating

7.CODING AND SOLUTIONING

Sign Up & Login:

```
const User = require("../models/user");
const bcrypt = require("bcryptjs");
const jwt = require("jsonwebtoken");

//---SIGN UP
exports.signUp = async (req, res) => {
  //Checking if user already exist
  const existEmail = await checkIfUserExist(req.body.email);

  //Check is account method is google
  if (existEmail) {
    if (req.body.accountMethod === "Google") {
      //redirect to sign in
      return this.signIn(req, res);
    } else {
      return res.status(500).send("Email already exist");
    }
  }

  const salt = await bcrypt.genSalt(10);
  let hashedPassword = null;
  let user = null;
  if (req.body.accountMethod === "Google") {
    //Hashing Password
    hashedPassword = await bcrypt.hash("signedByGoogle", salt);
    //Registering new user
    user = new User({
      email: req.body.email,
      username: req.body.username,
      password: hashedPassword,
      accountMethod: req.body.accountMethod,
    });

    console.log("by google");
  } else {
    //Hashing Password
    hashedPassword = await bcrypt.hash(req.body.password, salt);
    //Registering new user
    user = new User({
      email: req.body.email,
      username: req.body.username,
      password: hashedPassword,
    });
    console.log("not by google");
  }
  console.log(hashedPassword);
  console.log(user);

  user
    .save()
    .then(() => {
      const token = jwt.sign(
        {
          _id: user._id,
          username: user.username,
          email: user.email,
          accountMethod: user.accountMethod,
        },
        process.env.SECRET_TOKEN
      );
      res.header("auth-token", token).send(token);
    })
    .catch((err) => res.status(500).send(err));
};

//---SIGN IN
exports.signIn = async (req, res) => {
  //Checking if user exist
  console.log(req.body);
  const user = await checkIfUserExist(req.body.email);
  if (!user) return res.status(500).send("Email not exist");

  let validPassword = null;
  if (req.body.accountMethod === "Google") {
    //Checking if password is correct
    validPassword = await bcrypt.compare("signedByGoogle", user.password);
  } else {
    //Checking if password is correct
    validPassword = await bcrypt.compare(req.body.password,
user.password);
  }

  if (!validPassword) return res.status(500).send("Invalid Password");

  //Create and assign a token
  const token = jwt.sign(
    {
      _id: user._id,
      username: user.username,
      email: user.email,
      isAdmin: user.isAdmin,
      accountMethod: user.accountMethod,
    },
    process.env.SECRET_TOKEN
  );
  return res.header("auth-token", token).send(token);
};

//---RESET PASSWORD
exports.resetPassword = async (req, res) => {
  console.log(req.body);
  const salt = await bcrypt.genSalt(10);
  const hashedPassword = await bcrypt.hash(req.body.password, salt);
  const user = await User.findOneAndUpdate(
    { email: req.body.email },
    { password: hashedPassword }
  );
  if (user) {
    return res.status(200).send("New Password is updated");
  } else {
    return res.status(500).send("Email is not valid");
  }
};

exports.authUser = async (req, res) => {
```

```javascript
  const user = await checkIfUserExist(req.params.email);
  console.log(user);
  if (user) {
    return res.status(200).send("Continue");
  } else {
    return res.status(200).send("No user found");
  }
};

exports.updateUsername = async (req, res) => {
  User.findByIdAndUpdate(req.body._id, { username: req.body.username
}).then(
    (user) => {
      const token = jwt.sign(
        { _id: user._id, username: user.username, email: user.email },
        process.env.SECRET_TOKEN
      );
      res.header("auth-token", token).send(token);
    }
  );
};

const checkIfUserExist = async (email) => {
  const user = await User.findOne({ email });
  return user;
};
```

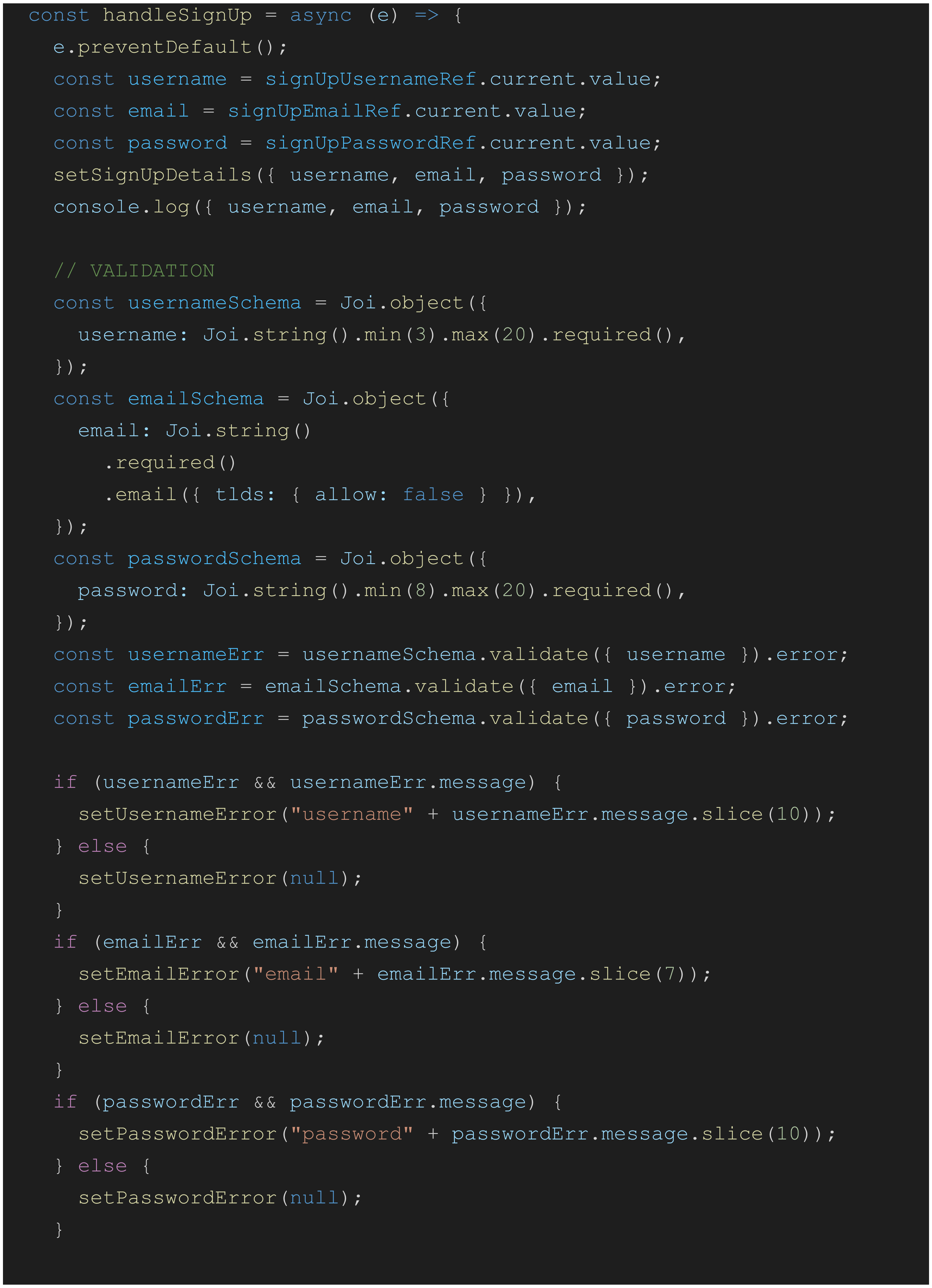
```javascript
import React, { useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import { GoogleButton } from "react-google-button";
import { UserAuth } from "../context/AuthContext";
import Button from "react-bootstrap/Button";
import Form from "react-bootstrap/Form";
import axios from "axios";
import Joi from "joi";
import toast from "react-hot-toast";
import jwtDecode from "jwt-decode";
import loginBg from "../assets/loginBg.jpg"
```

```
const SignIn = () => {
  const { userState, googleSignIn } = UserAuth();

  const [user, setUser] = userState;

  const [emailError, setEmailError] = useState();
  const [passwordError, setPasswordError] = useState();

  const signInEmailRef = React.createRef();
  const signInPasswordRef = React.createRef();

  const navigate = useNavigate();

  const handleSignIn = async (e) => {
    e.preventDefault();
    const email = signInEmailRef.current.value;
    const password = signInPasswordRef.current.value;
    console.log({ email, password });

    // VALIDATION
    const emailSchema = Joi.object({
      email: Joi.string()
        .required()
        .email({ tlds: { allow: false } }),
    });
    const passwordSchema = Joi.object({
      password: Joi.string().min(8).max(20).required(),
    });

    const emailErr = emailSchema.validate({ email }).error;
    const passwordErr = passwordSchema.validate({ password }).error;

    if (emailErr && emailErr.message) {
      setEmailError("email" + emailErr.message.slice(7));
    } else {
      setEmailError(null);
    }
    if (passwordErr && passwordErr.message) {
      setPasswordError("password" + passwordErr.message.slice(10));
    } else {
```

```
    setPasswordError(null);
  }

  if (!emailErr && !passwordErr) {
    await axios
      .post("http://localhost:8000/auth/sign-in", { email, password })
      .then((res) => {
        sessionStorage.setItem("token", res.data);
        console.log(res.data);
        setUser(jwtDecode(res.data));
        toast.success(`Welcome ${jwtDecode(res.data).username}`);
        navigate("/");
      })
      .catch((err) => {
        console.log(err.response.data);
        if (err.response.data === "Invalid Password") {
          setPasswordError(err.response.data);
        } else {
          setEmailError(err.response.data);
        }
      });
  }
};

const handleGoogleSignIn = async () => {
  try {
    await googleSignIn();
  } catch (error) {
    console.log(error);
  }
};
return (
  <div style ={{backgroundImage: `url(${loginBg})`,
  backgroundSize: 'cover',
  backgroundRepeat: 'no-repeat',
  width: '100%',
  height: '100vh'}}>
  <div
    style={{
      display: "flex",
        justifyContent: "center",
        alignItems: "center"

      }}
    >
      <div style={{ width: "400px",
        marginTop: "7%",
        backgroundColor:"rgb(114,214,203)",
        padding: "4%"}}>
        <h3 style={{textAlign: "center"}}>Sign In</h3>
        <br/>
        <Form>
          <Form.Group className="mb-3" controlId="formBasicEmail">
            <Form.Label>Email address</Form.Label>
            <Form.Control
              type="email"
              placeholder="Enter email"
              ref={signInEmailRef}
              defaultValue={"test@gmail.com"}
              style = {{backgroundColor:"whitesmoke"}}
            />
            {emailError && (
              <Form.Text className="text-danger">*
{emailError}</Form.Text>
            )}
          </Form.Group>

          <Form.Group className="mb-3" controlId="formBasicPassword">
            <Form.Label>Password</Form.Label>
            <Form.Control
              type="password"
              placeholder="Password"
              ref={signInPasswordRef}
              defaultValue={"12345678"}
              style = {{backgroundColor:"whitesmoke"}}
            />
            {passwordError && (
              <Form.Text className="text-danger">*
{passwordError}</Form.Text>
            )}
          </Form.Group>
          <Link
            to="/forgot-password"
            style={{
              textDecoration: "none",
              float: "right",
              marginTop: "-8px",
              marginBottom: "10px",
              color:"rgb(0, 0, 255)"
            }}
          >
            Forgot password?
          </Link>
          <Button
            type="submit"
            style={{ width:
"300px",backgroundColor:"rgb(63,59,62)",border:"none"}}
            onClick={handleSignIn}
          >
            Submit
          </Button>
          <br />

          <h6 className="text-center">or</h6>
        </Form>
        <GoogleButton
          onClick={handleGoogleSignIn}
          style={{ width: "300px" }}
        />
        <br />
        <p className="text-center">
          <Link
            to="/sign-up"
            style={{
              textDecoration: "none",
              marginTop: "-10px",
              marginBottom: "10px",
              color:"rgb(0, 0, 255)"
            }}
          >
```

```
            Doesn't have an account? Sign Up
          </Link>
        </p>
      </div>
    </div>
    </div>
  );
};

export default SignIn;
```

```
import React, { useRef, useState } from "react";
import { UserAuth } from "../context/AuthContext";
import { Link, useNavigate } from "react-router-dom";
import Button from "react-bootstrap/Button";
import Form from "react-bootstrap/Form";
import emailjs from "@emailjs/browser";
import Joi from "joi";
import toast from "react-hot-toast";
import axios from "axios";
import signupBg from "../assets/signupBg.jpg"

const SignUp = () => {
  const { otpState, signUpDetailsState } = UserAuth();

  const [otp, setOtp] = otpState;
  const [signUpDetails, setSignUpDetails] = signUpDetailsState;

  const [usernameError, setUsernameError] = useState();
  const [emailError, setEmailError] = useState();
  const [passwordError, setPasswordError] = useState();

  const signUpUsernameRef = React.createRef();
  const signUpEmailRef = React.createRef();
  const signUpPasswordRef = React.createRef();

  const form = useRef();
  const navigate = useNavigate();
```



```javascript
const handleSignUp = async (e) => {
  e.preventDefault();
  const username = signUpUsernameRef.current.value;
  const email = signUpEmailRef.current.value;
  const password = signUpPasswordRef.current.value;
  setSignUpDetails({ username, email, password });
  console.log({ username, email, password });

  // VALIDATION
  const usernameSchema = Joi.object({
    username: Joi.string().min(3).max(20).required(),
  });
  const emailSchema = Joi.object({
    email: Joi.string()
      .required()
      .email({ tlds: { allow: false } }),
  });
  const passwordSchema = Joi.object({
    password: Joi.string().min(8).max(20).required(),
  });
  const usernameErr = usernameSchema.validate({ username }).error;
  const emailErr = emailSchema.validate({ email }).error;
  const passwordErr = passwordSchema.validate({ password }).error;

  if (usernameErr && usernameErr.message) {
    setUsernameError("username" + usernameErr.message.slice(10));
  } else {
    setUsernameError(null);
  }
  if (emailErr && emailErr.message) {
    setEmailError("email" + emailErr.message.slice(7));
  } else {
    setEmailError(null);
  }
  if (passwordErr && passwordErr.message) {
    setPasswordError("password" + passwordErr.message.slice(10));
  } else {
    setPasswordError(null);
  }
```

```
    if (!usernameErr && !emailErr && !passwordErr) {
      const res = await axios.get(
        `http://localhost:8000/auth/auth-user/${email}`
      );
      if (res.data === "Continue") {
        return setEmailError("Email already exist");
        // return console.log(res.data + "Enter another account");
      }
      const localOtp = Math.floor(Math.random() * 1000000 + 1);
      setOtp(localOtp);
      console.log(localOtp);
      document.getElementById("otp").value =
        "Your email confirmation OTP is " + localOtp;
      emailjs
        .sendForm(
          "service_o0h67jj",
          "template_fldwtll",
          form.current,
          "A-XUgIf_QWxYJpLpm"
        )
        .then(
          (result) => {
            console.log(result.text);
            navigate("/otp-auth");
            // toast.success("Otp has sent to our email", {
            //   duration: 6000,
            // });
          },
          (error) => {
            console.log(error.text);
          }
        );
    }
  };

  return (
    <div style = {{backgroundImage: `url(${signupBg})`,
    backgroundSize: 'cover',
    backgroundRepeat: 'no-repeat',
    width: '100%',
    height: '100vh'}}>
    <div
      style={{
        display: "flex",
        justifyContent: "center",
        alignItems: "center",
      }}
    >
      <div style={{  width: "400px",
        marginTop: "7%",
        backgroundColor:"rgb(114,214,203)",
        padding: "4%"  }}>
        <h3 style={{textAlign:"center"}}>Sign Up</h3>
        <br/>
        <Form ref={form}>
          <Form.Control type="hidden" name="message" id="otp" />
          <Form.Group className="mb-3" controlId="formBasicEmail">
            <Form.Label>Username</Form.Label>
            <Form.Control
              type="text"
              placeholder="Enter name"
              ref={signUpUsernameRef}
              name="username"
              style = {{backgroundColor:"whitesmoke"}}
            />
            {usernameError && (
              <Form.Text className="text-danger">*
{usernameError}</Form.Text>
            )}
          </Form.Group>
          <Form.Group className="mb-3" controlId="formBasicEmail">
            <Form.Label>Email address</Form.Label>
            <Form.Control
              type="email"
              placeholder="Enter email"
              ref={signUpEmailRef}
              name="user_email"
              style = {{backgroundColor:"whitesmoke"}}
            />
            {emailError && (
              <Form.Text className="text-danger">*
{emailError}</Form.Text>
            )}
          </Form.Group>
          <Form.Group className="mb-4" controlId="formBasicPassword">
            <Form.Label>Password</Form.Label>
            <Form.Control
              type="password"
              placeholder="Password"
              ref={signUpPasswordRef}
              style = {{backgroundColor:"whitesmoke"}}
            />
            {passwordError && (
              <Form.Text className="text-danger">*
{passwordError}</Form.Text>
            )}
          </Form.Group>
          <Button
            variant="dark"
            type="submit"
            style={{ width: "300px",backgroundColor:"rgb(63,59,62)" }}
            onClick={handleSignUp}
          >
            Submit
          </Button>
          <br />
        </Form>
        <br />

        <p className="text-center">
          <Link
            to="/sign-in"
            style={{
              textDecoration: "none",
              marginTop: "-10px",
              marginBottom: "10px",
            }}
          >
            Already have an account? Sign In
          </Link>
        </p>
      </div>
    </div>
    </div>
  );
};

export default SignUp;
```

```
import React, { useState, useEffect } from "react";
import { UserAuth } from "../context/AuthContext";
import Button from "react-bootstrap/Button";
import Form from "react-bootstrap/Form";
import toast from "react-hot-toast";
import axios from "axios";
import jwtDecode from "jwt-decode";
import { useNavigate } from "react-router-dom";

const UserDetailsForm = () => {
  const { userState, signUpDetailsState } = UserAuth();
  const [user, setUser] = userState;
  const [signUpDetails, setSignUpDetails] = signUpDetailsState;

  useEffect(() => {
    const fetchUser = () => {
      const userToken = sessionStorage.getItem("token");
      userToken ? setUser(jwtDecode(userToken)) : setUser(null);
    };
    fetchUser();
  }, []);

  const ageRef = React.createRef();
  const sexRef = React.createRef();
  const chestPainRef = React.createRef();
  const bpRef = React.createRef();
  const cholesterolRef = React.createRef();
  const fbsRef = React.createRef();
  const ekgRef = React.createRef();
  const maxHrRef = React.createRef();
  const exerciseAnginaRef = React.createRef();
  const stDepressionRef = React.createRef();
  const slopeOfStRef = React.createRef();
  const numberOfVesselsRef = React.createRef();
  const thalliumRef = React.createRef();

  const [ageError, setAgeError] = useState();
  const [sexError, setSexError] = useState();
  const [chestPainError, setChestPainError] = useState();
  const [bpError, setBpError] = useState();
  const [cholesterolError, setCholesterolError] = useState();
  const [fbsError, setFbsError] = useState();
  const [ekgError, setEkgError] = useState();
  const [maxHrError, setMaxHrError] = useState();
  const [exerciseAnginaError, setExerciseAnginaError] = useState();
  const [stDepressionError, setStDepressionError] = useState();
  const [slopeOfStError, setSlopeOfStError] = useState();
  const [numberOfVesselsError, setNumberOfVesselsError] = useState();
  const [thalliumError, setThalliumError] = useState();

  const [buttonName, setButtonName] = useState("Calculate result");
  let [loading, setLoading] = useState(false);

  const navigate = useNavigate();

  const handleUserDetails = async (e) => {
    e.preventDefault();
    const age = Number(ageRef.current.value);
    const sex = Number(sexRef.current.value);
    const chestPain = Number(chestPainRef.current.value);
    const bp = Number(bpRef.current.value);
    const cholesterol = Number(cholesterolRef.current.value);
    const fbs = Number(fbsRef.current.value);
    const ekg = Number(ekgRef.current.value);
    const maxHr = Number(maxHrRef.current.value);
    const exerciseAngina = Number(exerciseAnginaRef.current.value);
    const stDepression = Number(stDepressionRef.current.value);
    const slopeOfSt = Number(slopeOfStRef.current.value);
    const numberOfVessels = Number(numberOfVesselsRef.current.value);
    const thallium = Number(thalliumRef.current.value);
```

```
// console.log(typeof sex);
// console.log(stDepression);
let error1 = 0;
let error2 = 0;
let error3 = 0;
let error4 = 0;
let error5 = 0;
let error6 = 0;
let error7 = 0;
let error8 = 0;
let error9 = 0;
let error10 = 0;
let error11 = 0;
let error12 = 0;
let error13 = 0;

if (!age || age < 1) {
  setAgeError("Enter valid age");
  error1 = 1;
} else {
  setAgeError(null);
  error1 = 0;
}
if (sex < 0 || sex > 1) {
  setSexError("Select 0 or 1");
  error2 = 1;
} else {
  setSexError(null);
  error2 = 0;
}
if (chestPain < 1 || chestPain > 4) {
  setChestPainError("Value should be between 1 to 4");
  error3 = 1;
} else {
  setChestPainError(null);
  error3 = 0;
}
if (!bp || bp < 94 || bp > 200) {
  setBpError("Value should be between 94 to 200");
```

```
    error4 = 1;
  } else {
    setBpError(null);
    error4 = 0;
  }
  if (!cholesterol || cholesterol < 100 || cholesterol > 600) {
    setCholesterolError("Value should be between 100 to 600");
    error5 = 1;
  } else {
    setCholesterolError(null);
    error5 = 0;
  }
  if (fbs < 0 || fbs > 1) {
    setFbsError("Value should be 0 or 1");
    error6 = 1;
  } else {
    setFbsError(null);
    error6 = 0;
  }
  if (ekg < 0 || ekg > 2) {
    setEkgError("Value should between 0 to 2");
    error7 = 1;
  } else {
    setEkgError(null);
    error7 = 0;
  }
  if (!maxHr || maxHr < 70 || maxHr > 250) {
    setMaxHrError("Value should be between 70 to 250");
    error8 = 1;
  } else {
    setMaxHrError(null);
    error8 = 0;
  }
  if (exerciseAngina < 0 || exerciseAngina > 1) {
    setExerciseAnginaError("Value should be 0 or 1");
    error9 = 1;
  } else {
    setExerciseAnginaError(null);
    error9 = 0;
  }
```

```
    if (stDepression < 0 || stDepression > 10) {
      setStDepressionError("Value should be between 0 to 10");
      error10 = 1;
    } else {
      setStDepressionError(null);
      error10 = 0;
    }
    if (slopeOfSt < 1 || slopeOfSt > 3) {
      setSlopeOfStError("Value should be between 1 to 3");
      error11 = 1;
    } else {
      setSlopeOfStError(null);
      error11 = 0;
    }
    if (numberOfVessels < 0 || numberOfVessels > 3) {
      setNumberOfVesselsError("Value should be between 1 to 3");
      error12 = 1;
    } else {
      setNumberOfVesselsError(null);
      error12 = 0;
    }
    if (thallium < 3 || thallium > 7) {
      setThalliumError("Value should be between 3 to 7");
      error13 = 1;
    } else {
      setThalliumError(null);
      error13 = 0;
    }
    if (
      !error1 &&
      !error2 &&
      !error3 &&
      !error4 &&
      !error5 &&
      !error6 &&
      !error7 &&
      !error8 &&
      !error9 &&
      !error10 &&
      !error11 &&
      !error12 &&
      !error13
    ) {
      console.log("hello");
      setButtonName("Your result is loading...");
      setLoading(true);
      const newUser = {
        ...signUpDetails,
        userDetails: {
          age,
          sex,
          chestPain,
          bp,
          cholesterol,
          fbs,
          ekg,
          maxHr,
          exerciseAngina,
          stDepression,
          slopeOfSt,
          numberOfVessels,
          thallium,
        },
      };
      console.log(newUser);
      if (!user) {
        await axios
          .post("http://localhost:8000/auth/sign-up", newUser)
          .then((res) => {
            sessionStorage.setItem("token", res.data.token);
            sessionStorage.setItem("justSignedUp", null);
            console.log(res.data);
            setUser(jwtDecode(res.data.token));
            toast(res.data.answer, {
              duration: 6000,
            });
            navigate("/");
          })
          .catch((err) => console.log(err));
      } else {
        await axios
          .post(`http://localhost:8000/user/add-user-details/${user._id}`,
{
            age,
            sex,
            chestPain,
            bp,
            cholesterol,
            fbs,
            ekg,
            maxHr,
            exerciseAngina,
            stDepression,
            slopeOfSt,
            numberOfVessels,
            thallium,
          })
          .then((res) => {
            console.log(res.data);
            toast.success(res.data, {
              duration: "6000",
            });
            setButtonName("Add details");
            setLoading(false);
            // navigate("/");
          })
          .catch((err) => {
            console.log(err);
          });
      }
    }
  };

  return (
    <div>


      <div
        style={{
          // width: "90%",
          backgroundColor: "rgb(114,214,203)",

        }}
      ><div className="text-center">
      <h3 style={{color: "black",padding:"1%"}}>
        <i>User Details</i>
      </h3>
    </div>
        <div
          style={{
            display: "flex",
            flexWrap: "wrap",
            justifyContent: "center",
            marginTop: "30px",
            color:"black", fontSize:"bold",
          }}
        >
          <div style={{ width: "300px", padding: "15px" }}>
            <Form.Group className="mb-3" controlId="formBasicEmail">
              <Form.Label>Age</Form.Label>
              <Form.Control
                type="number"
                placeholder="Enter age"
                ref={ageRef}
              />
              {ageError && (
                <i className="text-info">* {ageError}</i>
              )}
            </Form.Group>

            <Form.Group className="mb-3" controlId="formBasicEmail">
              <Form.Label>Sex</Form.Label>
              <Form.Select aria-label="Default select example"
ref={sexRef}>
                <option value="-1" hidden>
                  Select sex
                </option>
                <option value="0">0</option>
                <option value="1">1</option>
              </Form.Select>
              {sexError && (
                <i className="text-info">* {sexError}</i>
              )}
            </Form.Group>

            <Form.Group className="mb-3" controlId="formBasicEmail">
              <Form.Label>Chest Pain</Form.Label>
              <Form.Select
                aria-label="Default select example"
                ref={chestPainRef}
              >
                <option value="-1" hidden>
                  Select chest pain
                </option>
                <option value="1">1</option>
                <option value="2">2</option>
                <option value="3">3</option>
                <option value="4">4</option>
              </Form.Select>
              {chestPainError && (
                <i className="text-info">
                  * {chestPainError}
                </i>
              )}
            </Form.Group>

            <Form.Group className="mb-3" controlId="formBasicEmail">
              <Form.Label>BP</Form.Label>
              <Form.Control type="number" placeholder="Enter BP"
ref={bpRef} />
              {bpError && (
                <i className="text-info">* {bpError}</i>
              )}
            </Form.Group>
            <Form.Group className="mb-3" controlId="formBasicEmail">
              <Form.Label>Cholesterol</Form.Label>
              <Form.Control
                type="number"
                placeholder="Enter cholesterol"
                ref={cholesterolRef}
              />
              {cholesterolError && (
                <i className="text-info">
                  * {cholesterolError}
                </i>
              )}
            </Form.Group>
          </div>

          <div style={{ width: "300px", padding: "15px" }}>
            <Form.Group className="mb-3" controlId="formBasicEmail">
              <Form.Label>Fbs</Form.Label>
              <Form.Select aria-label="Default select example"
ref={fbsRef}>
                <option value="-1" hidden>
                  Select fbs
                </option>
                <option value="0">0</option>
                <option value="1">1</option>
              </Form.Select>
              {fbsError && (
                <i className="text-info">* {fbsError}</i>
              )}
            </Form.Group>

            <Form.Group className="mb-3" controlId="formBasicEmail">
              <Form.Label>Ekg</Form.Label>
              <Form.Select aria-label="Default select example"
ref={ekgRef}>
                <option value="-1" hidden>
                  Select ekg
                </option>
                <option value="0">0</option>
                <option value="1">1</option>
                <option value="2">2</option>
              </Form.Select>
              {ekgError && (
                <i className="text-info">* {ekgError}</i>
              )}
            </Form.Group>
```

```
      <Form.Group className="mb-3" controlId="formBasicEmail">
        <Form.Label>Max HR</Form.Label>
        <Form.Control
          type="number"
          placeholder="Enter Max HR"
          ref={maxHrRef}
        />
        {maxHrError && (
          <i className="text-info">* {maxHrError}</i>
        )}
      </Form.Group>
      <Form.Group className="mb-3" controlId="formBasicEmail">
        <Form.Label>Exercise Angina</Form.Label>
        <Form.Select
          aria-label="Default select example"
          ref={exerciseAnginaRef}
        >
          <option value="-1" hidden>
            Select exercise angina
          </option>
          <option value="0">0</option>
          <option value="1">1</option>
        </Form.Select>
        {exerciseAnginaError && (
          <i className="text-info">
            * {exerciseAnginaError}
          </i>
        )}
      </Form.Group>

      <Form.Group className="mb-3" controlId="formBasicEmail">
        <Form.Label>ST Depression</Form.Label>
        <Form.Control
          type="number"
          placeholder="Enter ST depression"
          defaultValue={0}
          ref={stDepressionRef}
        />
        {stDepressionError && (
          <i className="text-info">
            * {stDepressionError}
          </i>
        )}
      </Form.Group>
    </div>

    <div style={{ width: "300px", padding: "15px" }}>
      <Form.Group className="mb-3" controlId="formBasicEmail">
        <Form.Label>Slope of ST</Form.Label>
        <Form.Select
          aria-label="Default select example"
          ref={slopeOfStRef}
        >
          <option value="-1" hidden>
            Select slope of ST
          </option>
          <option value="1">1</option>
          <option value="2">2</option>
          <option value="3">3</option>
        </Form.Select>
        {slopeOfStError && (
          <i className="text-info">
            * {slopeOfStError}
          </i>
        )}
      </Form.Group>

      <Form.Group className="mb-3" controlId="formBasicEmail">
        <Form.Label>Number of vessels fluro</Form.Label>
        <Form.Select
          aria-label="Default select example"
          ref={numberOfVesselsRef}
        >
          <option value="-1" hidden>
            Select number of vessels fluro
          </option>
          <option value="1">1</option>
          <option value="2">2</option>
          <option value="3">3</option>
```

```
              </Form.Select>
              {numberOfVesselsError && (
                <i className="text-info">
                  * {numberOfVesselsError}
                </i>
              )}
            </Form.Group>

            <Form.Group className="mb-3" controlId="formBasicEmail">
              <Form.Label>Thallium</Form.Label>
              <Form.Select
                aria-label="Default select example"
                ref={thalliumRef}
              >
                <option value="-1" hidden>
                  Select thallium
                </option>
                <option value="3">3</option>
                <option value="4">4</option>
                <option value="5">5</option>
                <option value="6">6</option>
                <option value="7">7</option>
              </Form.Select>
              {thalliumError && (
                <i className="text-info">* {thalliumError}</i>
              )}
            </Form.Group>
          </div>
        </div>

        <div
          style={{
            display: "flex",
            justifyContent: "center",
            alignItems: "center",
          }}
        >
          <Button
            variant="success"
            type="submit"
            style={{ width: "300px", marginBottom: "76px" }}
            onClick={handleUserDetails}
          >
            {buttonName}

          </Button>
        </div>
      </div>
    </div>
  );
};

export default UserDetailsForm;
```

8. TESTING

8.1 Test Cases:

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LOGIN													
LoginPage_TC_OO1	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on My account button		1.Enter URL and click go 2.Click on My Account dropdown button 3.Verify login/Singup popup displayed or not	http://localhost:3000	Login/Signup popup should display	Working as expected	Pass				Ravichandran B
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in Login/Signup popup		1.Enter URL and click go 2.Click on My Account dropdown button 3.Verify login/Singup popup with	http://localhost:3000	Application should show below UI elements: a.email text box b.password text box c.Login button with	Working as expected	Pass				Pavithran M

					below UI elements : a.email text box b.password text box c.Login button d.New customer ? Create account link e.Last password ? Recovery password link		orange colour d.New customer ? Create account link e.Last password ? Recovery password link						
LoginPage_TC_OO3	Functional	Home page	Verify user is able to log into application with Valid credentials		1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter valid password	Username: alpha@gmail.com password: Alpha123	User should navigate to user account homepage	Working as expected	Pass				Prem nath S

					in password text box 5.Click on login button								
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3.Enter InValid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: alpha@gmail password: Alpha123	Application should show 'Incorrect email or password' validation message.	Working as expected	Pass				Premnath S
LoginPage_TC_OO5	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account	Username: alpha@gmail.com password: Alpha1236786	Application should show 'Incorrect email or password' validation message.	Working as expected	Pass				Ranjith K S

					dropdown button 3.Enter Valid username/email in Email text box 4.Enter Invalid password in password text box 5.Click on login button	86786876876							
LoginPage_TC_OO6	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3.Enter InValid username/email in Email text box 4.Enter Invalid password in password text box 5.Click on	Username: alpha password: Alpha123678686786876876	Application should show 'Incorrect email or password' validation message.	Working as expected	Pass				Ranjith K S

					login button								
SIGNUP													
Signup Page_TC_OO1	Functional	Home page	Verify user is able to signup into application with Valid credentials		1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on signup button 6.Enter the OTP received on the registered email	Username: alpha1@gmail.com password: Alpha123	User should navigate to fill up the user details	Working as expected	Pass				Ravichandran B

Signup Page_TC_OO2	Functional	Home page	Verify user is able to signup into application with Valid credentials		1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3.Enter Existing username/email in Email text box 4.Enter valid password in password text box 5.Click on signup button	Username: alpha1@gmail.com password: Alpha123	Application should show ' Email/user already exist' validation message.	Working as expected	Pass				Pavithran M
Signup Page_TC_OO3	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3.Enter InValid username/email	Username: alpha1gmail.com password: Alpha123	Application should show 'Email must contain @ ' validation message.	Working as expected	Pass				Premnath S

					in Email text box 4.Enter valid password in password text box 5.Click on login button								
Signup Page_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter Invalid password in password text box 5.Click on login button	Username: alpha@gmail.com password: Alp12	Application should show 'Password must be atleast 8 characters' validation message.	Working as expected	Pass				Ravichandran B

SignupPage_TC_OO5	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3.Enter InValid username/email in Email text box 4.Enter Invalid password in password text box 5.Click on login button	Username: alpha@gmail.com password: Alpha123	Application should show 'Email must contain @ ' and 'Password must be atleast 8 characters' validation message	Working as expected	Pass				Ravichandran B

9. PERFORMANCE METRICS

Performance Metrics:

Performance metrics of the machine learning algorithm is evaluated using confusion matrix.

10. ADVANTAGES AND DISADVANTAGES

Advantages:

- Increase the Accuracy for effective Heart disease Diagnosis
- Reduce the time Complexity of Doctors
- Cost Effective for patients
- Predicts the likelihood of patients getting heart disease
- Predicts people with cardiovascular disease by extracting the patient's medical history that leads to fatal health.
- Improves health care services
- Initial Setup and maintenance cost is reduced

11. CONCLUSION

Conclusion:

Early detection of cardiovascular diseases can help high-risk patients make decisions about lifestyle changes that will lessen complications, which can be a significant advancement in the field of medicine. In order to forecast heart disease, this project used computer algorithms.

12. FUTURE SCOPE

Future Scope:

There were numerous methods and steps involved in the model building, validation, and deployment. Future research will focus on predicting the target attribute while streamlining the processes and phases. In comparison to other prediction models that have already been developed and are covered in the literature review, the model's accuracy is also somewhat lower. The pipeline layout and algorithm selection process will need to be adjusted in order to increase accuracy.

APPENDIX

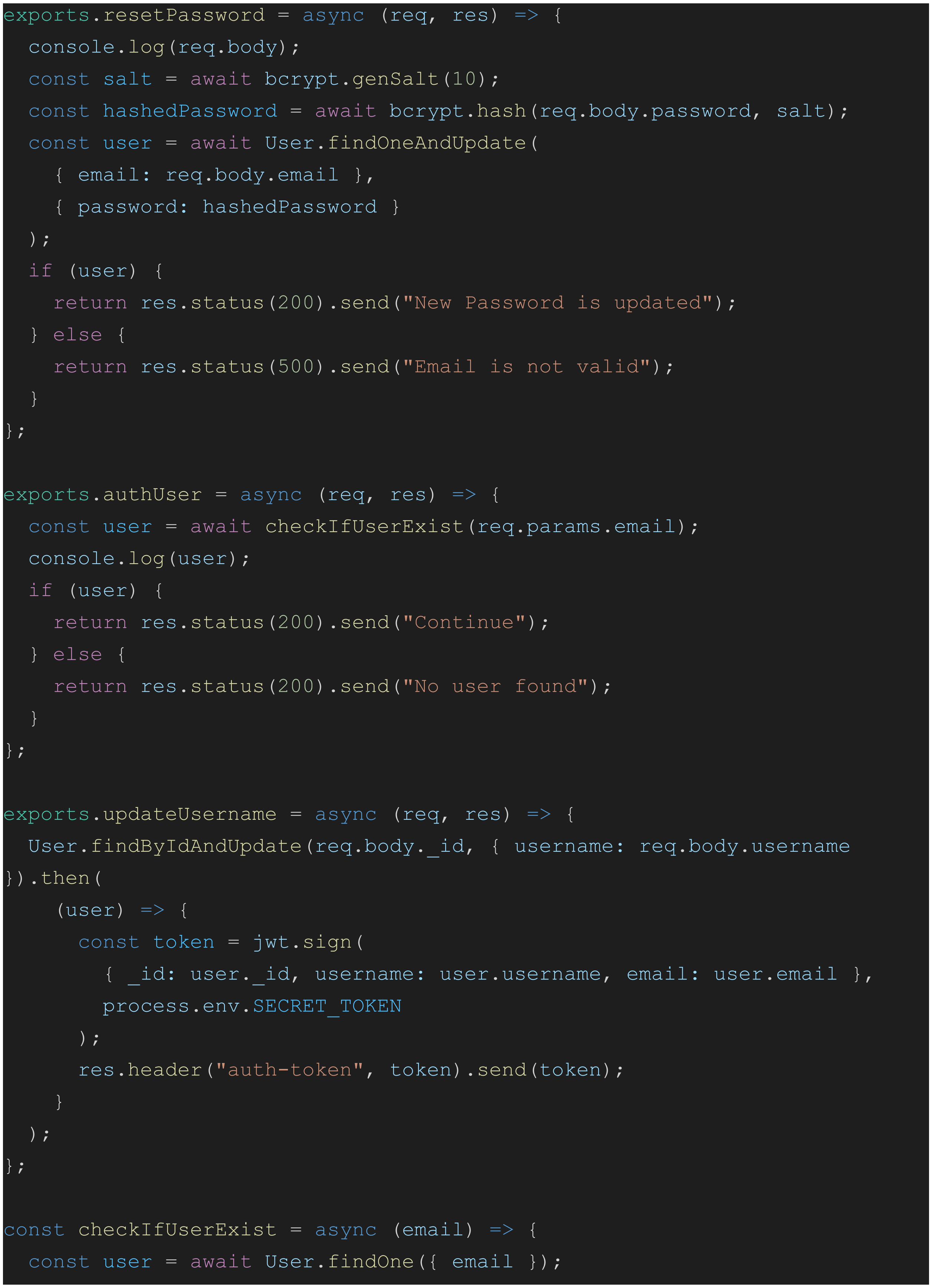
Source Code:

```
const Feedback = require("../models/feedback");
const Query = require("../models/query");
const User = require("../models/user");

exports.addFeedback = (req, res) => {
  console.log(req.body);
  const newFeedback = new Feedback({
    ...req.body,
  });
  newFeedback
    .save()
    .then(() => res.status(200).send("Thanks for your Feedback"))
    .catch((err) => res.status(500).send(err));
};

exports.fetchUserFeedbacks = async (req, res) => {
  const { id } = req.params;

  // console.log(id);
  await Feedback.find({ userId: id })
    .then((feedbacks) => {
      res.status(200).send(feedbacks);
    })
    .catch((err) => {
      res.status(500).send(err);
    });
};

exports.deleteFeedback = async (req, res) => {
  console.log(req.body);
  const feedback = await Feedback.findById(req.body._id);
  console.log(feedback.userId, req.body.userId);
  if (feedback.userId == req.body.userId) {
    Feedback.findByIdAndDelete(req.body._id)
      .then(() => {
        res.status(200).send("Your feedback has deleted");
      })
      .catch((err) => {
        res.status(500).send(err);
      });
  } else {
    res.status(500).send("You can't delete someone's feedback");
  }
};

exports.fetchUserQueries = async (req, res) => {
  const { id } = req.params;

  // console.log(id);
  await Query.find({ userId: id })
    .then((queries) => {
      res.status(200).send(queries);
    })
    .catch((err) => {
      res.status(500).send(err);
    });
};

exports.deleteQuery = async (req, res) => {
  console.log(req.body);
  const query = await Query.findById(req.body._id);
  console.log(query.userId, req.body.userId);
  if (query.userId == req.body.userId) {
    Query.findByIdAndDelete(req.body._id)
      .then(() => {
        res.status(200).send("Your query has deleted");
      })
      .catch((err) => {
        res.status(500).send(err);
      });
  } else {
    res.status(500).send("You can't delete someone's query");
  }
};
```

```
exports.addUserDetails = async(req, res) => {
  console.log(req.body);
  await User.findByIdAndUpdate(req.params.id, {userDetails:
{...req.body}})
  .then(() => res.status(200).send("Updated"))
  .catch(err => res.status(500).send(err));
}

exports.fetchUserDetails = async(req, res) => {
  console.log(req.body);
  await User.findById(req.params.id)
  .then(user => res.status(200).send(user.userDetails))
  .catch(err => res.status(500).send(err));
}


// exports.deleteAccount = (req, res) => {
//   const { userId } = req.body;
//   User.findByIdAndDelete(userId)
//     .then(() => res.status(200).send("Account has been deleted
successfully"))
//     .catch((err) => res.status(500).send(err));
// };


exports.addUserDetails = async (req, res) => {
  console.log({ ...req.body });
  await User.findByIdAndUpdate(req.params.id, { userDetails: { ...req.body
} })
    .then(() => {
      const { spawn } = require("child_process");
      const pyProg = spawn("python", [
        "D:/WebDev Projects/IBM-APP/server/controllers/predict.py",
        req.body.age,
        req.body.sex,
        req.body.chestPain,
        req.body.bp,
        req.body.cholesterol,
        req.body.fbs,
        req.body.ekg,
        req.body.maxHr,
        req.body.exerciseAngina,
        req.body.stDepression,
        req.body.slopeOfSt,
        req.body.numberOfVessels,
        req.body.thallium,
      ]);
      let answer = "";

      pyProg.stdout.on("data", function (data) {
        console.log(data.toString());
        answer += data.toString();
      });

      pyProg.stdout.on("end", function () {
        res.status(200).send(answer);
      });
    })
    .catch((err) => res.status(500).send(err));
};
```

```
const User = require("../models/user");
const bcrypt = require("bcryptjs");
const jwt = require("jsonwebtoken");

//---SIGN UP
exports.signUp = async (req, res) => {
  //Checking if user already exist
  const existEmail = await checkIfUserExist(req.body.email);

  //Check is account method is google
  if (existEmail) {
    if (req.body.accountMethod === "Google") {
      //redirect to sign in
      return this.signIn(req, res);
    } else {
      return res.status(500).send("Email already exist");
    }
  }
```

```
const salt = await bcrypt.genSalt(10);
let hashedPassword = null;
let user = null;
if (req.body.accountMethod === "Google") {
  //Hashing Password
  hashedPassword = await bcrypt.hash("signedByGoogle", salt);
  //Registering new user
  user = new User({
    email: req.body.email,
    username: req.body.username,
    password: hashedPassword,
    accountMethod: req.body.accountMethod,
  });

  console.log("by google");
} else {
  //Hashing Password
  hashedPassword = await bcrypt.hash(req.body.password, salt);
  //Registering new user
  user = new User({
    email: req.body.email,
    username: req.body.username,
    password: hashedPassword,
  });
  console.log("not by google");
}
console.log(hashedPassword);
console.log(user);

user
  .save()
  .then(() => {
    const token = jwt.sign(
      {
        _id: user._id,
        username: user.username,
        email: user.email,
        accountMethod: user.accountMethod,
      },
      process.env.SECRET_TOKEN
      );
      res.header("auth-token", token).send(token);
    })
    .catch((err) => res.status(500).send(err));
};

//---SIGN IN
exports.signIn = async (req, res) => {
  //Checking if user exist
  console.log(req.body);
  const user = await checkIfUserExist(req.body.email);
  if (!user) return res.status(500).send("Email not exist");

  let validPassword = null;
  if (req.body.accountMethod === "Google") {
    //Checking if password is correct
    validPassword = await bcrypt.compare("signedByGoogle", user.password);
  } else {
    //Checking if password is correct
    validPassword = await bcrypt.compare(req.body.password,
user.password);
  }

  if (!validPassword) return res.status(500).send("Invalid Password");

  //Create and assign a token
  const token = jwt.sign(
    {
      _id: user._id,
      username: user.username,
      email: user.email,
      isAdmin: user.isAdmin,
      accountMethod: user.accountMethod,
    },
    process.env.SECRET_TOKEN
  );
  return res.header("auth-token", token).send(token);
};

//---RESET PASSWORD
```



```
exports.resetPassword = async (req, res) => {
  console.log(req.body);
  const salt = await bcrypt.genSalt(10);
  const hashedPassword = await bcrypt.hash(req.body.password, salt);
  const user = await User.findOneAndUpdate(
    { email: req.body.email },
    { password: hashedPassword }
  );
  if (user) {
    return res.status(200).send("New Password is updated");
  } else {
    return res.status(500).send("Email is not valid");
  }
};

exports.authUser = async (req, res) => {
  const user = await checkIfUserExist(req.params.email);
  console.log(user);
  if (user) {
    return res.status(200).send("Continue");
  } else {
    return res.status(200).send("No user found");
  }
};

exports.updateUsername = async (req, res) => {
  User.findByIdAndUpdate(req.body._id, { username: req.body.username
}).then(
    (user) => {
      const token = jwt.sign(
        { _id: user._id, username: user.username, email: user.email },
        process.env.SECRET_TOKEN
      );
      res.header("auth-token", token).send(token);
    }
  );
};

const checkIfUserExist = async (email) => {
  const user = await User.findOne({ email });
  return user;
};
```

Github: https://github.com/IBM-EPBL/IBM-Project-10590-1659191911