Early Prediction of Diabetes Using Machine Learning

Submitted by

Teertho Kamol Goshami Likhon
ID: 201-35-3032
Department of Software Engineering
Daffodil International University

Submitted to

Mohammad Khaled Sohel
Assistant Professor
Department of Software Engineering
Daffodil International University

This project report is submitted to fulfil the requirements for the Bachelor of Science in Software Engineering degree.

APPROVAL

This project titled on "**Early Prediction of Diabetes Using Machine Learning**", submitted by **Teertho Kamol Goshami Likhon (ID: 201-35-3032)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS

Fazla Elahe 10.7.24

--- **Chairman**

Dr. Md. Fazla Elahe
Assistant Professor & Associate Head
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

--- **Internal Examiner 1**

Tapushe Rabaya Toma
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

--- **Internal Examiner 2**

Khalid Been Md. Badruzzaman Biplob
Lecturer (Senior Scale)
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

--- **External Examiner**

Md Mostafiz Khan
Managing Director
Tecognize Solutions Limited

DECLARATION

I confirm that this project has been completed with the guidance of **Mohammad Khaled Sohel, Assistant Professor** in the Department of Software Engineering at **Daffodil International University.**

Submitted by:

Teertho Kamol Goshami Likhon
ID: 201-35-3032
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Supervised by:

Mohammad Khaled Sohel
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

ACKNOWLEDGMENT

I am first grateful to the Divine Being for guiding me through the successful completion of this project. My sincere thanks go to **Mohammad Khaled Sohel, Assistant Professor** in the **Software Engineering Department**, for his invaluable guidance and support. I also appreciate the contributions and enthusiasm of all those who assisted with the project, as their input was crucial to its success. I am thankful to the professors in the **Software Engineering Department** for their encouragement. Lastly, I deeply appreciate my parents for their constant prayers and support, which have been a driving force throughout this journey.

ABSTRACT

This project focuses on leveraging machine learning techniques to develop a predictive model for the early detection of diabetes. With the growing prevalence of diabetes globally, timely diagnosis plays a crucial role in managing the condition and preventing its associated complications. The study utilises a dataset containing relevant health parameters such as glucose levels, BMI, age, and blood pressure, sourced from diverse demographic groups.

Implemented within the Google Colab environment, various machine learning algorithms, including logistic regression, decision trees, and support vector machines, are employed to analyse the dataset and construct predictive models. Feature selection and engineering techniques are applied to enhance model performance and interpretability. Furthermore, the project explores the integration of deep learning methodologies for more nuanced pattern recognition.

Model performance evaluation involves rigorous testing and validation. Additionally, the models are assessed for their ability to provide interpretable insights into the factors contributing to diabetes risk.

The anticipated outcome of this research is the development of a robust and accurate predictive tool capable of identifying individuals at risk of diabetes at an early stage. Such a tool has the potential to facilitate proactive interventions and personalised healthcare strategies, ultimately contributing to improved patient outcomes and healthcare management efficiency.

DEDICATION

I dedicate this project to my supervisor, esteemed teachers, and my beloved parents, as well as to those who hold a special place in my heart. Their patience, support, understanding, and love have been instrumental in helping me achieve this milestone.

Table of Contents

Chapter 7: Conclusion

Chapter 1: Introduction

1.1 Project Overview

Greetings from GlucoGuard, Your Guardian Angel for Managing Your Blood Sugar. GlucoGuard is a cutting-edge program created to assist people in properly controlling and tracking their blood glucose levels. Cutting-edge machine learning algorithms are used by GlucoGuard to seamlessly interface with glucose monitoring devices and provide individualized insights into how nutrition, exercise, and daily habits affect blood sugar levels. This clever tool delivers predictive analytics in addition to real-time glucose monitoring, predicting possible variations and recommending preventative actions. The intuitive GlucoGuard interface makes it easier to read data and promotes a better knowledge of how different lifestyle factors affect different people. Take control of your glucose management journey with GlucoGuard, which helps you make educated choices for a more balanced and healthy way of living. With confidence, welcome the era of glucose monitoring that lies ahead.

1.2 Project Background

The rising prevalence of diabetes necessitates advanced tools for effective blood glucose management. Traditional monitoring methods often need more comprehensive insights and predictive capabilities, leading to challenges in maintaining optimal health. GlucoGuard addresses these issues by integrating cutting-edge machine-learning algorithms with glucose monitoring devices. GlucoGuard provides real-time tracking, personalized insights, and predictive analytics, enabling users to understand how diet, exercise, and daily habits impact their blood sugar levels. Its user-friendly interface simplifies data interpretation, empowering users to make informed decisions and adopt proactive measures. By anticipating potential fluctuations, GlucoGuard helps prevent hypo- and hyperglycemic episodes, significantly enhancing the quality of life for individuals with diabetes. In summary, GlucoGuard revolutionizes diabetes management by offering a reliable, intelligent, and comprehensive solution, guiding users toward a healthier and more balanced lifestyle.

1.3 Project Purpose

Seamless Integration: GlucoGuard effortlessly syncs with your glucose monitoring devices for real-time tracking.

Personalized Insights: Utilizes state-of-the-art machine learning algorithms to provide tailored insights into daily patterns, diet, and exercise impacts on blood sugar levels. Predictive analytics anticipates potential fluctuations and suggests proactive measures for effective glucose management.

User-Friendly Interface: Simplifies data interpretation, fostering a deeper understanding of individual responses to lifestyle factors.

Informed Decisions: Empowers users to make informed decisions for a healthier and more balanced lifestyle.

Trusted Companion: Be confident in your glucose management journey with GlucoGuard as your reliable and intelligent companion.

Optimal Well-being: Embrace the future of glucose monitoring, guiding you towards optimal well-being with confidence and ease.

1.4 Project Motivation

Prevalence: Rising global diabetes rates pose a significant health challenge.

Timely Detection: Traditional methods may delay identifying early signs, impacting interventions.

Machine Learning Opportunity: ML's potential for analyzing vast datasets can enhance early prediction.

Challenges: Developing accurate, accessible ML models for diabetes prediction requires addressing diverse risk factors and individual health profiles.

Goal: Innovate robust, interpretable ML solutions for early detection, enabling personalized management, and improving health outcomes.

1.5 Project Objective

- Provide real-time tracking of blood glucose levels through seamless integration with monitoring devices.
- Offer personalized insights into how diet, exercise, and daily habits impact glucose levels.
- Utilize predictive analytics to anticipate potential fluctuations and suggest preventative measures.
- Simplify data interpretation through a user-friendly interface.
- Empower users to make informed decisions for better diabetes management and overall well-being.

1.6 Project Schedule

+ Add　Expand all　Collapse all　Zoom in　Zoom out　Zoom to fit

ID	Name	Start Date	End Date	Duration	Color
1	Early Prediction of Diabetes Using Machine Learning	Jul 03, 2023	Jun 28, 2024	260 days	
2	Project Planing	Jul 03, 2023	Aug 31, 2023	44 days	
3	Analysis	Sep 01, 2023	Nov 30, 2023	65 days	
4	Design	Dec 01, 2023	Jan 31, 2024	44 days	
5	Development	Feb 01, 2024	May 03, 2024	67 days	
6	Testing & Integration	May 06, 2024	Jun 07, 2024	25 days	
7	Maintenance	Jun 07, 2024	Jun 28, 2024	16 days	

Figure 1: Project schedule date and time.

1.7 Gantt Chart

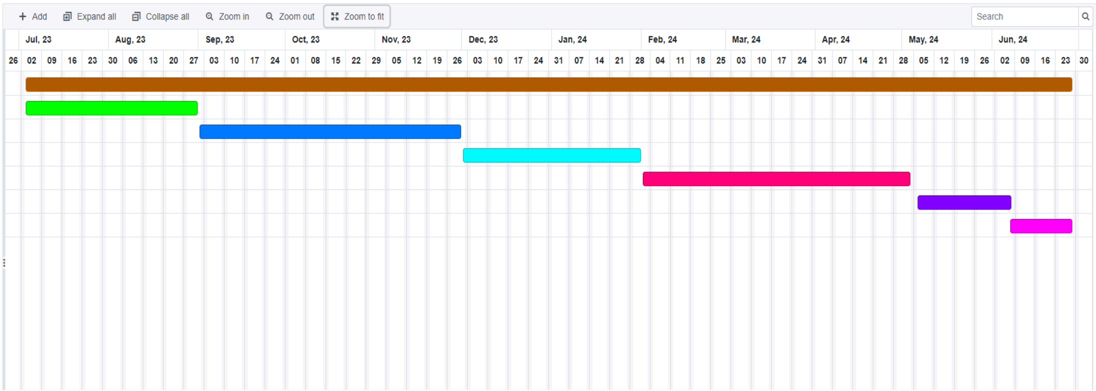


Figure 2: Gantt chart schedule.

1.8 Stakeholders:

1. User
2. Doctor
3. Admin

1.9 Literature Review:

Existing software for Android:

1. Diabetic diary
2. BloodPressureDB
3. MySugar Glucose

Existing software for PCs:

1. DietPower
2. Diabetes Pilot
3. Fitness Assistant
4. Health Tracker

Working Process:

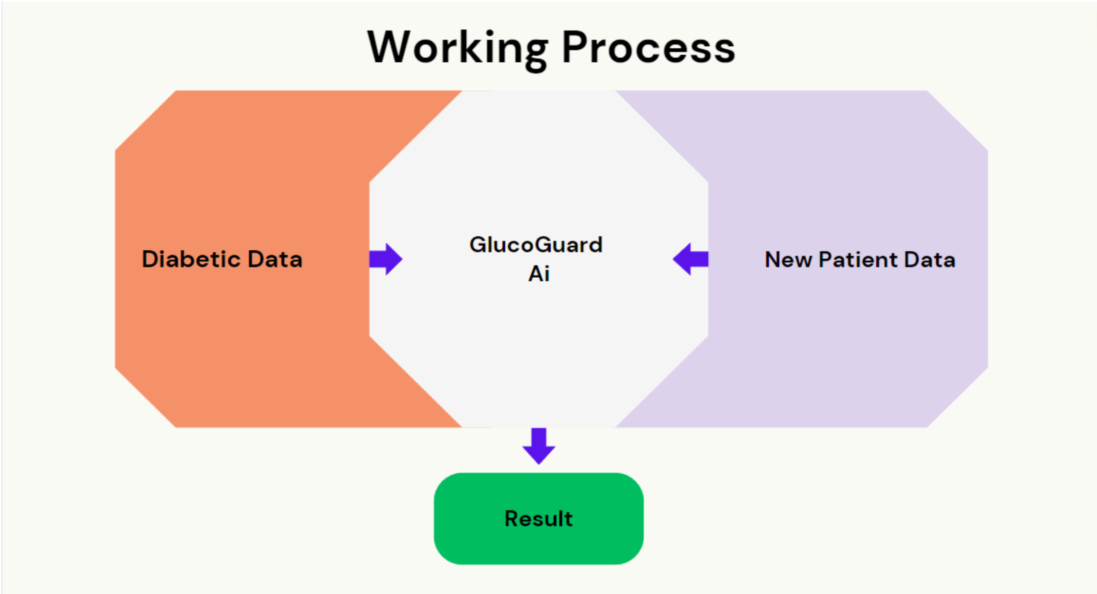


Figure 3: Working process of this project.

Chapter 2: Requirement Engineering

2.1 Functional Requirements:

1. Entry Patient Data
2. View Body Status
3. View Comparison
4. View Doctor
5. Remind Notification
6. Generate Report
7. Quick Health
8. Connect Device
9. Quick Support
10. Update Profile

2.2 Non-Functional Requirements:

1. Performance
2. Security
3. Scalability
4. Reliability
5. Usability

2.3 Software Requirements Specification (SRS):

2.3.1 Registration

FR-001	Registration
Description	The user registers with the system.
Stakeholder	User

2.3.2 Login

FR-002	Login
Description	The user logs in to the system.
Stakeholder	User

2.3.3 Entry Patient Data

FR-003	Entry Patient Data
Description	Enable users to input their health metrics, including blood sugar levels, weight, and dietary habits.
Stakeholder	User

2.3.4 View Body Status

FR-004	View Body Status
Description	Users can monitor and track various body metrics, such as weight, blood pressure, heart rate, and other relevant parameters.
Stakeholder	User

2.3.5 View Comparison

FR-005	View Comparison
Description	Users can compare their current body status with historical data or predefined benchmarks to track progress or identify deviations.
Stakeholder	User

2.3.6 View Doctor

FR-006	View Doctor
Description	Users can access a directory of healthcare professionals, view profiles, and schedule appointments.
Stakeholder	User

2.3.7 Remind Notification

FR-007	Remind Notification
Description	Users can set reminders for medication schedules, doctor appointments, or other healthcare-related tasks.

Stakeholder	User, Doctor

2.3.8 Generate Report

FR-008	Generate Report
Description	Users can generate comprehensive reports summarising patient data, body status, trends, and other relevant information.
Stakeholder	User, Doctor

2.3.9 Quick Health

FR-009	Quick Health
Description	Users can access quick health insights, tips, or recommendations based on their profile and current health status.
Stakeholder	User

2.4.1 Connect Device

FR-010	Connect Device
Description	Users can connect compatible healthcare devices such as fitness trackers, blood pressure monitors, or glucometers to sync data with the system.
Stakeholder	User

2.4.2 Quick Support

FR-011	Quick Support
Description	Users can access support resources, and FAQs, or contact customer support for assistance with system usage or healthcare-related queries.
Stakeholder	User, Doctor, Admin

2.4.3 Update Profile

FR-012	Update Profile
Description	Users have the ability to update and change the personal information, medical history, and preferences listed in their profiles.
Stakeholder	User, Doctor

2.4.4 Logout

FR-013	Logout
Description	Users and administrators log out of the system.
Stakeholder	User, Doctor, Admin

2.4 Feasibility Analysis

GlucoGuard is technically, operationally, economically, and legally feasible. With the integration of advanced machine learning techniques, accessible cloud-based resources, and strong market potential, the project is well-positioned to succeed and make a meaningful impact on diabetes management.

1. Technical Feasibility
2. Operational Feasibility
3. Economic Feasibility
4. Legal and Ethical Feasibility

2.5 Technical Feasibility

GlucoGuard leverages established machine-learning algorithms and integrates seamlessly with existing glucose monitoring devices. The use of Google Colab ensures scalable computing resources and access to a rich ecosystem of libraries for data analysis and machine learning. The project requires robust data collection and processing pipelines, which are achievable with current technology.

2.6 Operational Feasibility

GlucoGuard's user-friendly interface ensures ease of use for individuals with varying levels of technical expertise. Real-time tracking and personalized insights facilitate daily glucose management, making it operationally viable. The integration of predictive analytics provides actionable recommendations, enhancing user engagement and adherence to management plans. Training and support materials can be developed to assist users in effectively utilizing the platform.

2.7 Economic Feasibility

The economic feasibility of GlucoGuard involves evaluating the cost-effectiveness and financial sustainability of developing and deploying the tool. Initial investment costs include software development, machine learning model training, and integration with existing glucose monitoring devices. However, the potential for cost savings through improved diabetes management, reduced healthcare expenses, and fewer complications provides strong economic justification. Additionally, a subscription-based model or partnerships with healthcare providers could generate steady revenue streams, ensuring long-term financial viability.

2.8 Legal and Ethical Feasibility

The legal and ethical feasibility of GlucoGuard focuses on compliance with healthcare regulations and the ethical management of user data. Ensuring compliance with standards such as HIPAA (Health Insurance Portability and Accountability Act) is essential to protecting user privacy and securing sensitive health information. Ethically, the program must prioritize user consent, data transparency, and accuracy in its predictive analytics to maintain trust. By adhering to these legal and ethical guidelines, GlucoGuard can effectively safeguard user rights and foster a trustworthy relationship with its users.

2.9 Performance

The accuracy, dependability, user happiness, personalization, scalability, and user engagement of GlucoGuard are the key performance indicators. These two attributes demonstrate high precision in glucose monitoring and dependable machine-learning forecasts.

Predictive analytics: proactive suggestions and efficient foresight of changes in blood sugar levels.

User Engagement: Consistent use and simple data interpretation are encouraged via an intuitive interface.

Personalization: personalized advice based on personal information for successful glucose control.

Integration & Scalability: adaptability to a rising user base and smooth integration with a range of devices.

User happiness: Positive comments show a high level of user happiness and the usefulness of the tool.

3.1 Development Model:

1. Use Case Diagram
2. Use Case Description
3. Block Diagram
4. Activity Diagram
5. ER Diagram
6. Sequence Diagram
7. Class Diagram

3.2 Use Case Diagram:

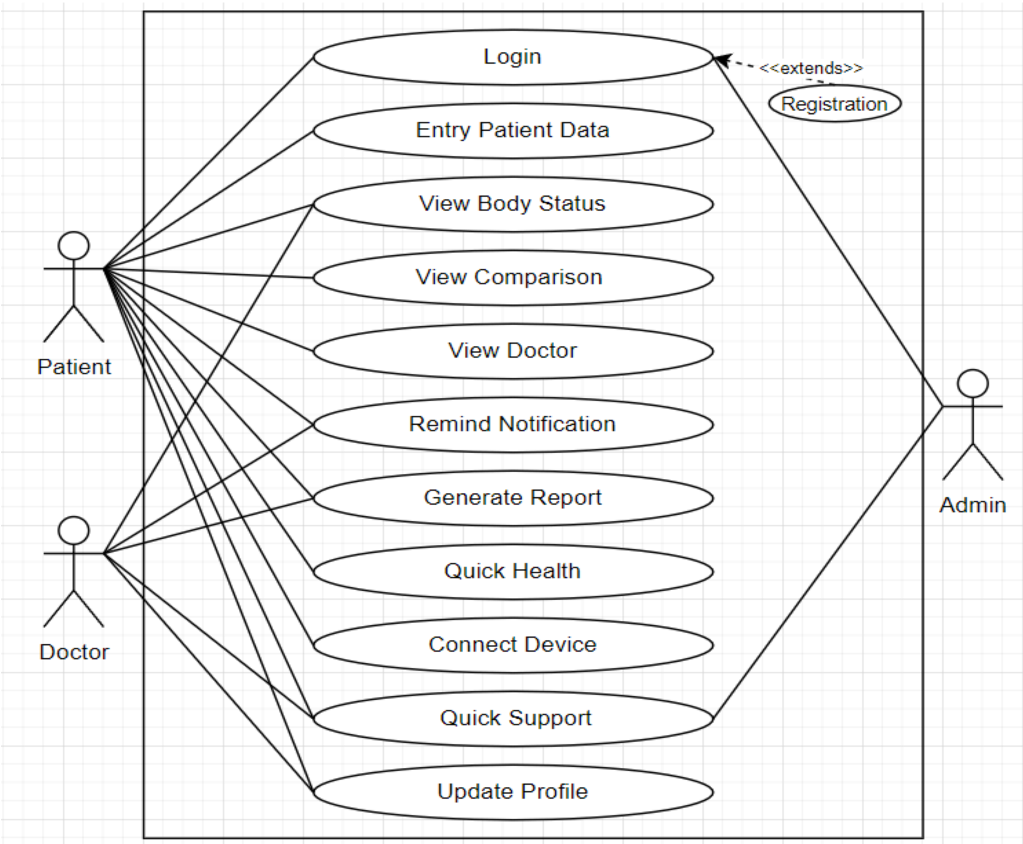


Figure 4: Use case diagram

3.3 Use Case Description:

3.3.1 Registration

Use case	Registration
Goal	Register to the system.
Precondition	N / A
Success End Condition	User registration is successful.
Fail End Condition	User registration is unsuccessful.
Primary Actor	Doctor, User, Admin
Secondary Actor	N/A
Tigger	A new user entered the system.
Description	Sign up as a new user in the system.
Alternative Flows	Mandatory fill in the missing information.

Quality Requirements	• Enter the first name • Enter the last name • Enter email • Enter phone number • Set the password • Click the Sign-Up

3.3.2 Log In

Use case	Log In
Goal	Log in to the system.
Precondition	The user must register in the system.
Success End Condition	The user login is successful.
Fail End Condition	The user login is unsuccessful.
Primary Actor	Doctor, User, Admin
Secondary Actor	N/A
Tigger	A registered user attempts to log in to the system.
Description	Authenticate and grant access to a registered user in the system.
Alternative Flows	Incorrect username or password entered: Prompt the user to re-enter credentials. Account locked after multiple failed attempts: Prompt the user to reset their password or contact support.
Quality Requirements	• Enter the username or email • Enter the password • Click the Login button • Redirect to the dashboard upon successful login • Display an error message upon unsuccessful login

3.3.3 Entry Patient Data

Use case	Entry Patient Data
Goal	Enter patient data into the system.
Precondition	The user must be authenticated and authorized.

Success End Condition	Patient data is successfully entered and saved.
Fail End Condition	Patient data entry is unsuccessful.
Primary Actor	Doctor, User, Admin
Secondary Actor	N/A
Tigger	A user initiates the process to enter new patient data.
Description	Enter and save patient information in the system.
Alternative Flows	Missing mandatory fields: Prompt the user to fill in the missing information. Invalid data format: prompt the user to correct the data.
Quality Requirements	● Enter patient's personal details ● Enter medical history ● Enter current health status ● Save the patient data

3.3.4 View Body Status

Use case	View body status
Goal	View the health status of a patient.
Precondition	The user must be authenticated and authorized.
Success End Condition	Body status is displayed correctly.
Fail End Condition	Body status fails to load.
Primary Actor	Doctor, User, Admin
Secondary Actor	N/A
Tigger	A user requests to view a patient's body status.
Description	Display the health status of a patient.
Alternative Flows	Data not available: Display an appropriate message.
Quality Requirements	● Select patient ● Display health metrics (e.g., weight, blood pressure, etc.)

3.3.5 View Comparison

Use case	View Comparison
Goal	Compare health data over time or between patients.
Precondition	The user must be authenticated and authorized.
Success End Condition	Comparison data is displayed correctly.
Fail End Condition	Comparison data fails to load.
Primary Actor	Doctor, User, Admin
Secondary Actor	N/A
Tigger	A user requests to compare health data.
Description	Display comparison of health data.
Alternative Flows	Data not available: Display an appropriate message.
Quality Requirements	● Select data points for comparison ● Display comparison chart or table

3.3.6 View Doctor

Use case	View Doctor
Goal	View the Doctor's information.
Precondition	User must be authenticated and authorized.
Success End Condition	Doctor's information is displayed.
Fail End Condition	Doctor's information fails to load.
Primary Actor	User, Admin
Secondary Actor	N/A
Tigger	A user requests to view doctor's information.
Description	Display the doctor's profile and contact details.
Alternative Flows	Data not available: Display an appropriate message.
Quality Requirements	● Select doctor

	• Display doctor's profile

3.3.7 Remind Notification

Use case	Remind Notification
Goal	Send reminder notifications to users.
Precondition	User must be authenticated and authorized.
Success End Condition	Reminder is sent successfully.
Fail End Condition	Reminder is not sent successfully.
Primary Actor	Admin
Secondary Actor	N/A
Tigger	A scheduled time or a user action triggers a reminder notification.
Description	Send notifications for reminders (appointments, medication, etc.).
Alternative Flows	Notification not delivered: Retry or log the failure.
Quality Requirements	• Set up reminder details • Send notification • Confirm delivery

3.3.8 Generate Report

Use case	Generate Report
Goal	Generate a report based on the system data.
Precondition	User must be authenticated and authorized.
Success End Condition	Report is generated successfully.
Fail End Condition	Report is not generated successfully.
Primary Actor	Admin, Doctor
Secondary Actor	N/A
Tigger	A user requests to generate a report.
Description	Create a detailed report based on the selected

	criteria.
Alternative Flows	Report generation error: Display an appropriate error message.
Quality Requirements	● Select report parameters ● Generate and display report ● Option to download or print report

3.3.9 Quick Health

Use case	Quick Health
Goal	Access quick health tips or information.
Precondition	User must be authenticated and authorized.
Success End Condition	Health information is displayed correctly.
Fail End Condition	Health information is not displayed correctly.
Primary Actor	User
Secondary Actor	N/A
Tigger	A user requests quick health information.
Description	Provide quick access to health tips and information.
Alternative Flows	Information not available: Display an appropriate message.
Quality Requirements	● Select health topic ● Display relevant information

3.4.1 Connect Device

Use case	Connect Device
Goal	Connect a health monitoring device to the system.
Precondition	User must be authenticated and authorized.

Success End Condition	Device is connected and data is syncing.
Fail End Condition	Device fails to connect.
Primary Actor	User, Doctor
Secondary Actor	N/A
Tigger	A user initiates the process to connect a device.
Description	Connect and sync data from a health monitoring device.
Alternative Flows	Connection error: Display an error message and retry option.
Quality Requirements	• Select device type • Pair device with system • Confirm data syncing

3.4.2 Quick Support

Use case	Quick Support
Goal	Provide quick support or assistance to users.
Precondition	User must be authenticated and authorized.
Success End Condition	Support is provided successfully.
Fail End Condition	Support is not provided successfully.
Primary Actor	User, Admin
Secondary Actor	N/A
Tigger	A user requests support.
Description	Offer quick support or troubleshooting assistance.
Alternative Flows	Support is not available: Display an appropriate message.
Quality Requirements	• Describe issue • Connect with support agent or display FAQs • Resolve issue or escalate

3.4.3 Update Profile

Use case	Update Profile
Goal	Update user profile information.
Precondition	User must be authenticated.
Success End Condition	Profile is updated successfully.
Fail End Condition	Profile update fails.
Primary Actor	User
Secondary Actor	N/A
Tigger	A user initiates the process to update their profile.
Description	Modify and save user profile information.
Alternative Flows	Invalid data format: Prompt user to correct the data.
Quality Requirements	• Enter new profile information • Save changes • Confirm update

3.4 Block Diagram:

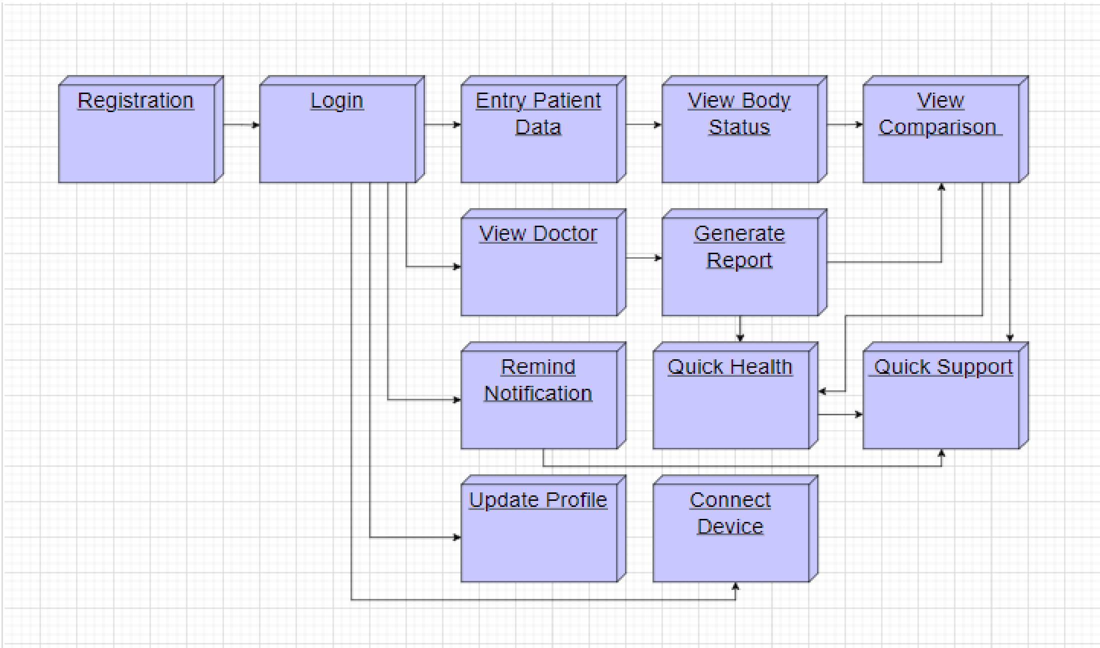
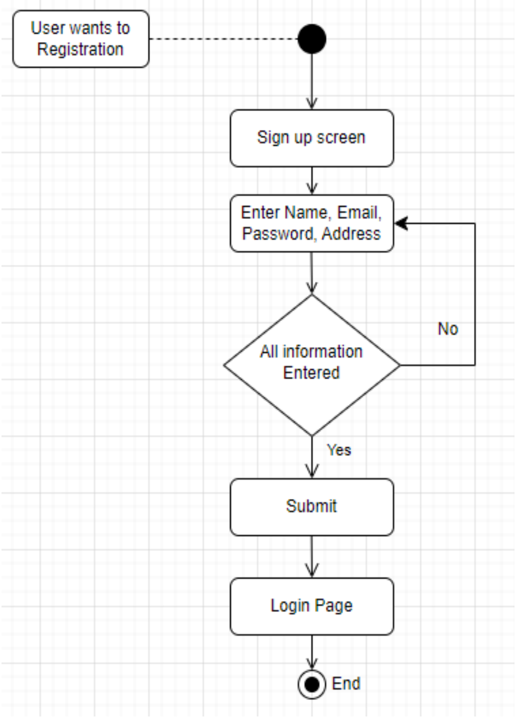


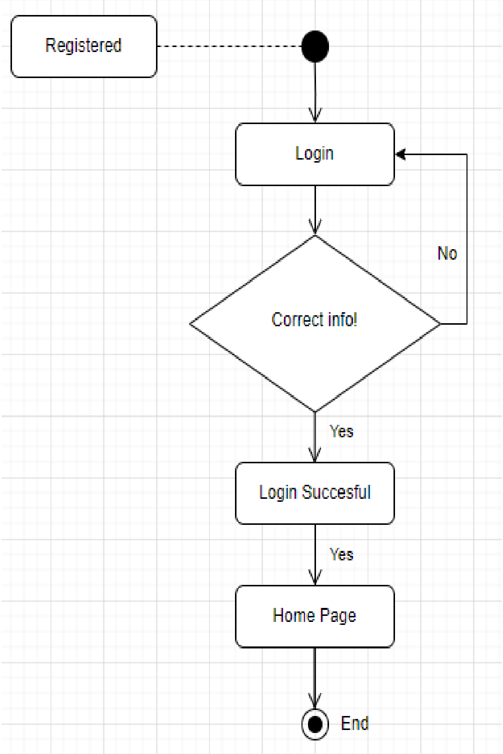
Figure 5: Block diagram of this project.

3.5 Activity Diagram:

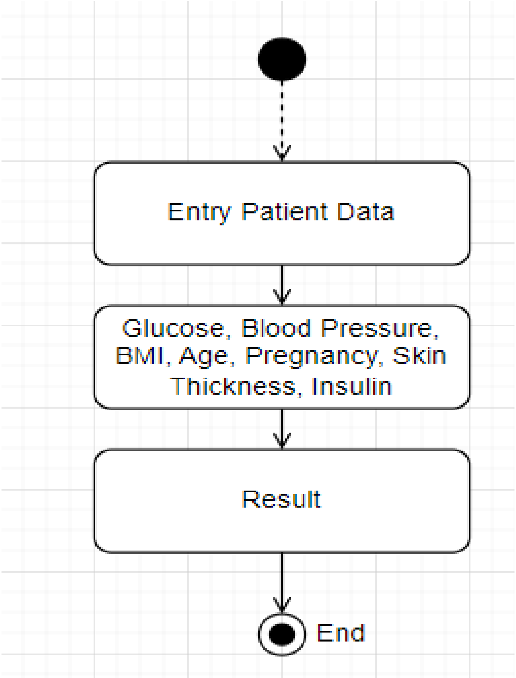
3.5.1 Registration



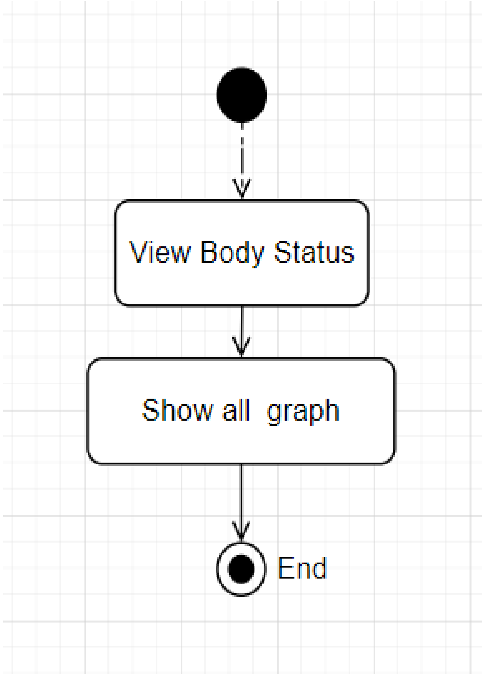
3.5.2 Login



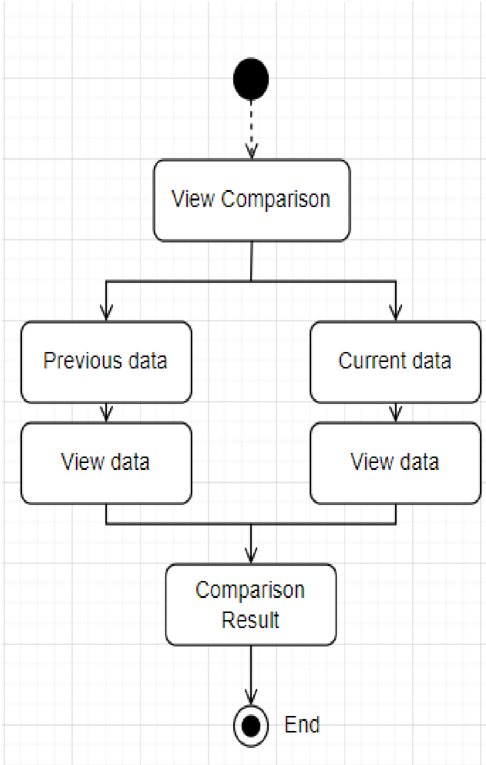
3.5.3 Entry Patient Data



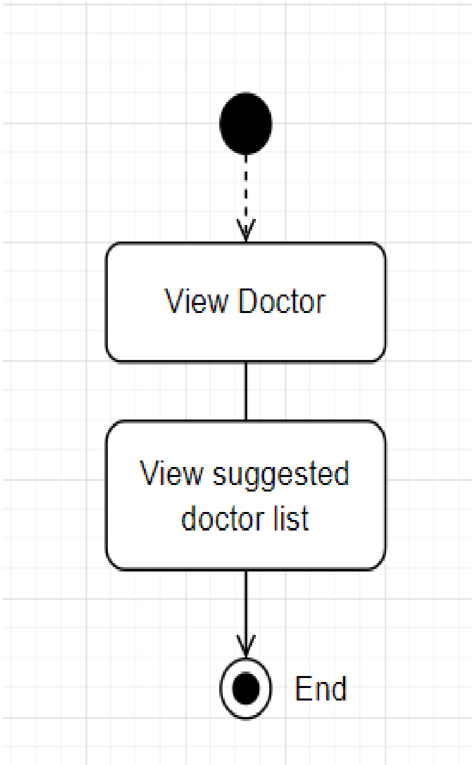
3.5.4 View Body Status



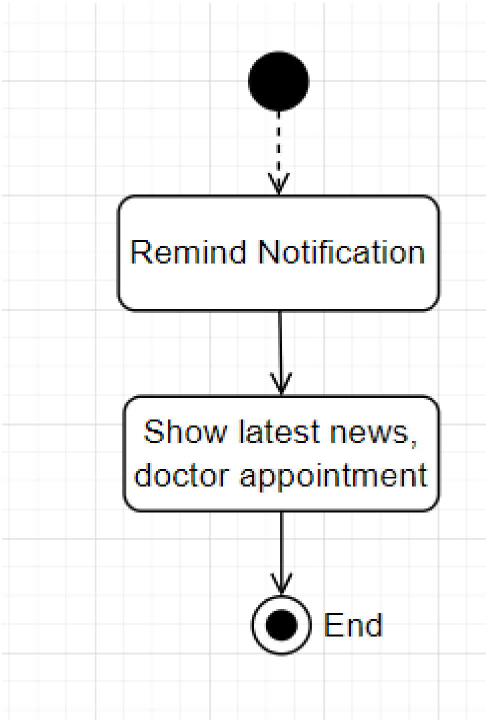
3.5.5 View Comparison



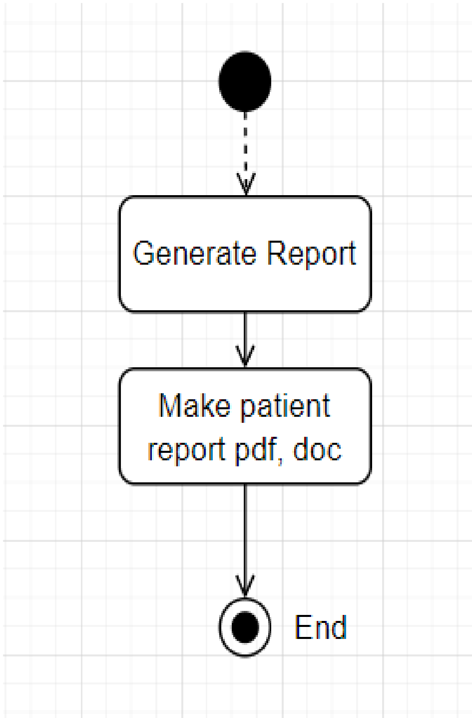
3.5.6 View Doctor



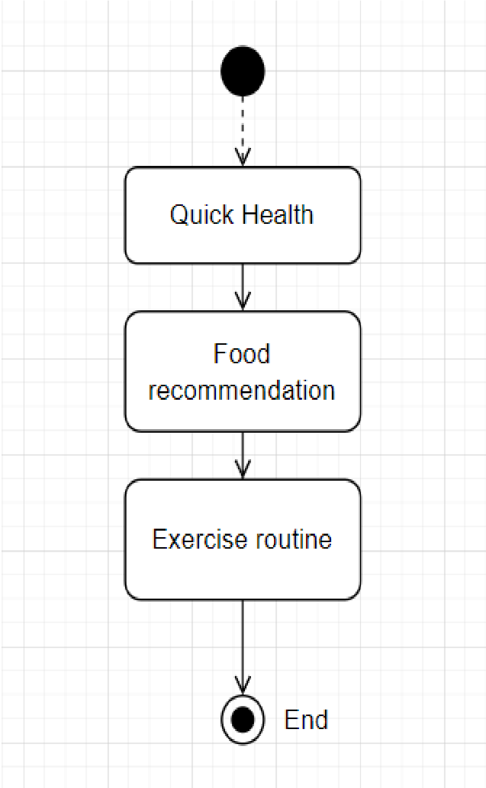
3.5.7 Remind Notification



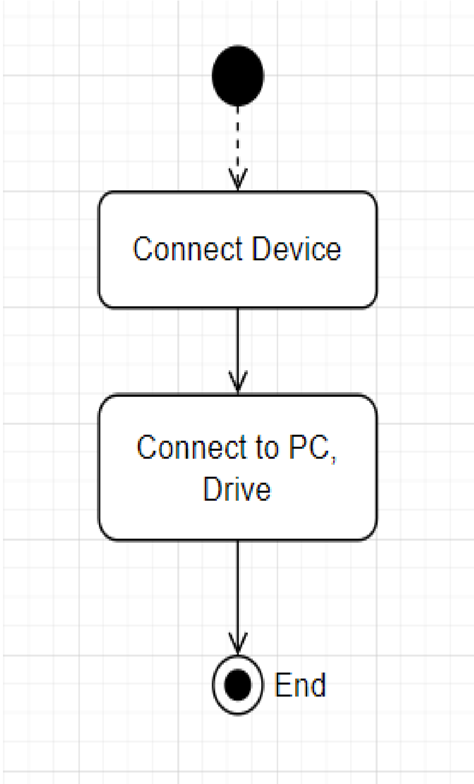
3.5.8 Generate Report



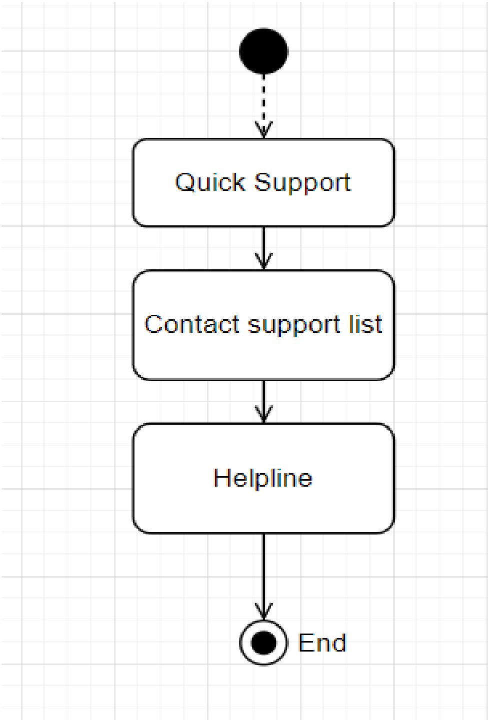
3.5.9 Quick Health



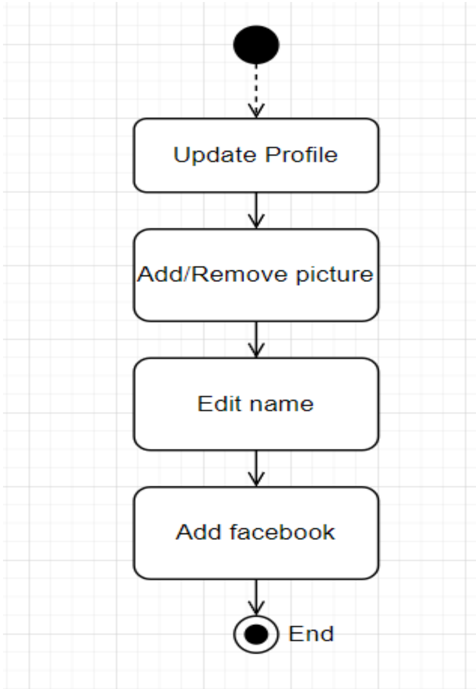
3.6.1 Connect Device



3.6.2 Quick Support



3.6.3 Update Profile



3.6 Entity Relationship Diagram (ERD):

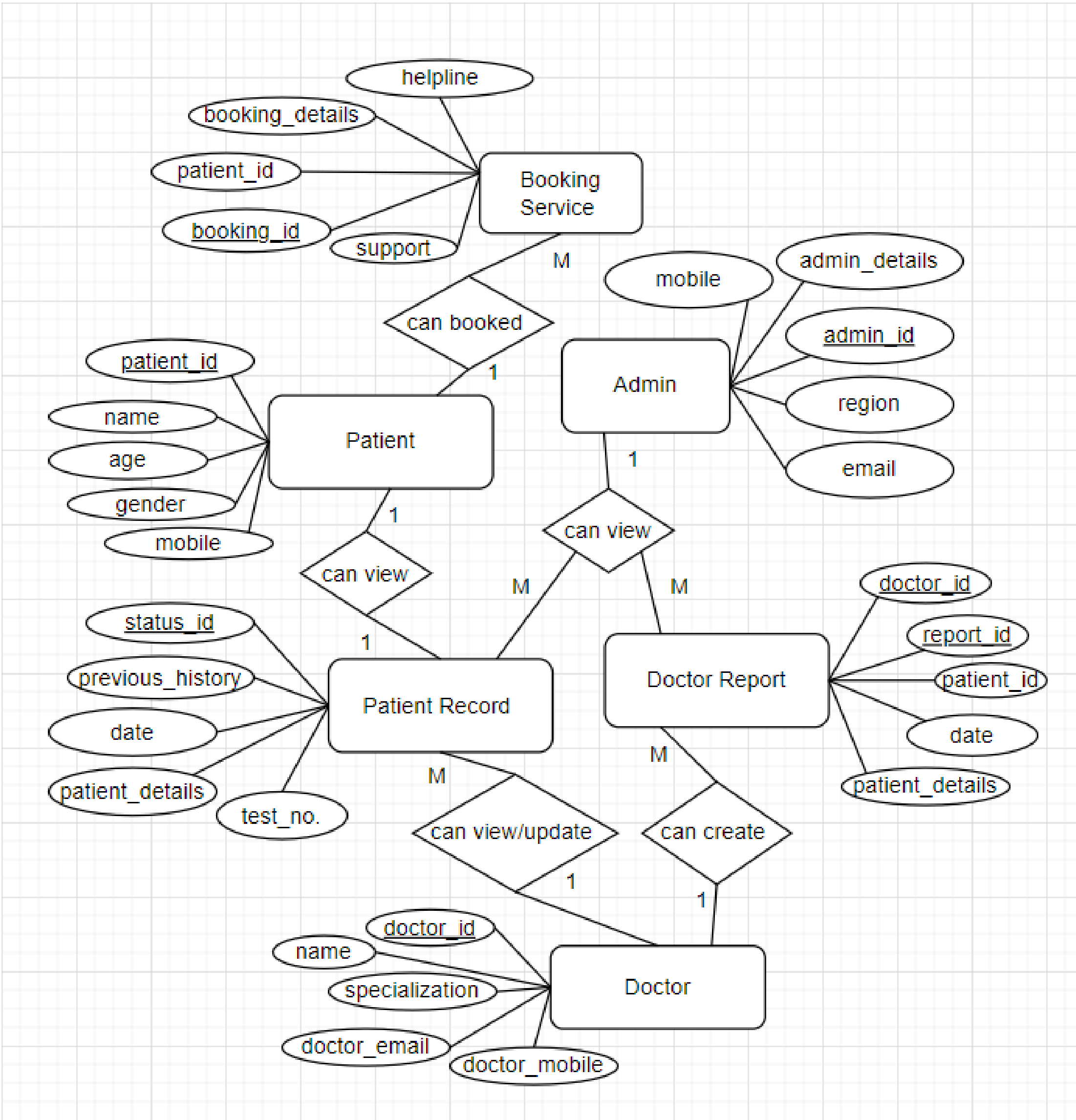


Figure 6: ERD of this project.

3.7 Sequence Diagram:

3.7.1 Registration Page:

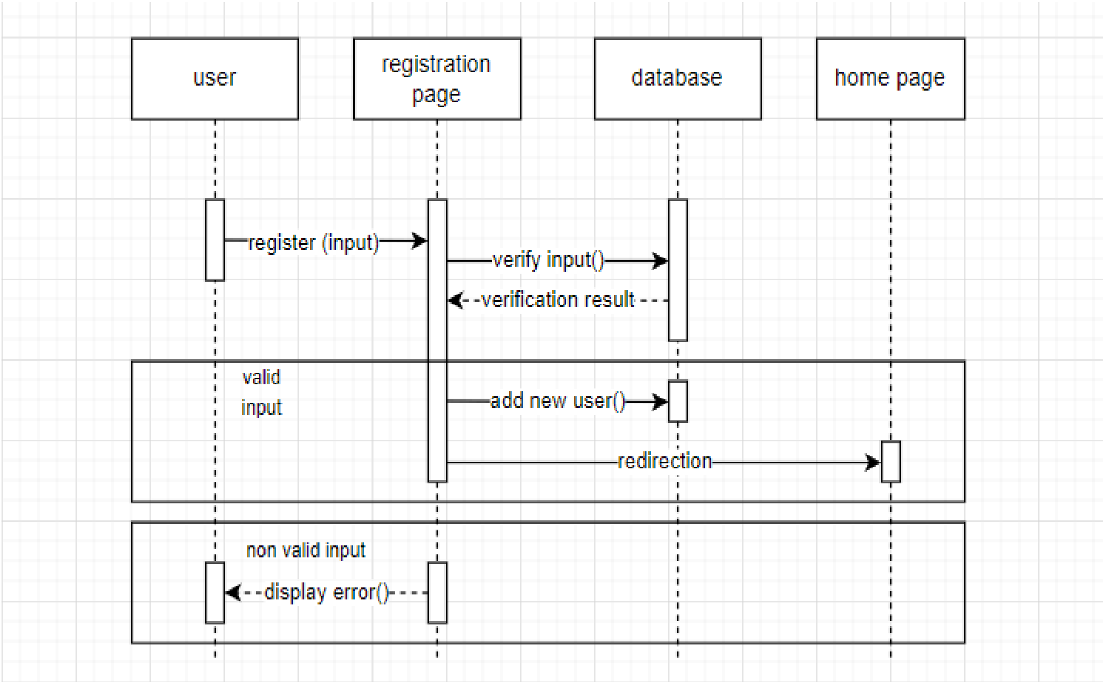


Figure 7: Registration page

3.7.2 Home Page:

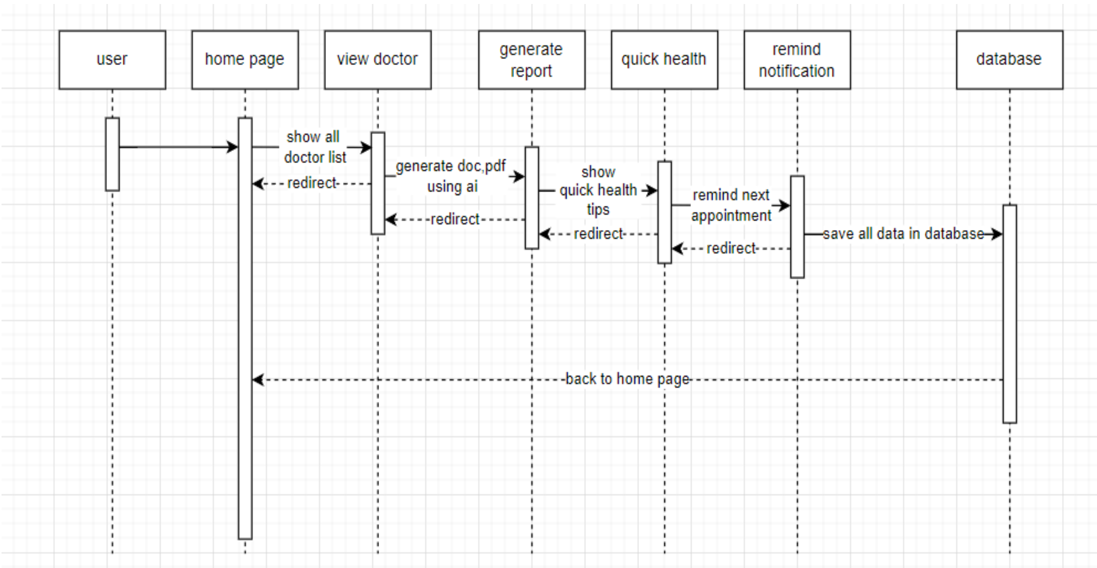


Figure 8: Home page

3.7.3 Update Profile:

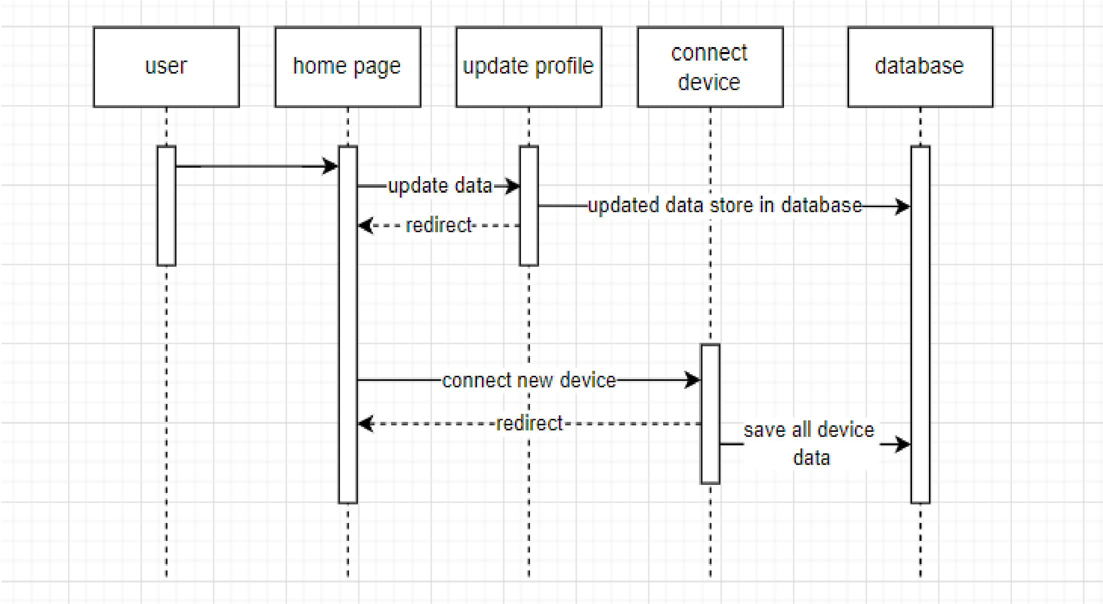


Figure 9: Update profile

3.8 Class Diagram:

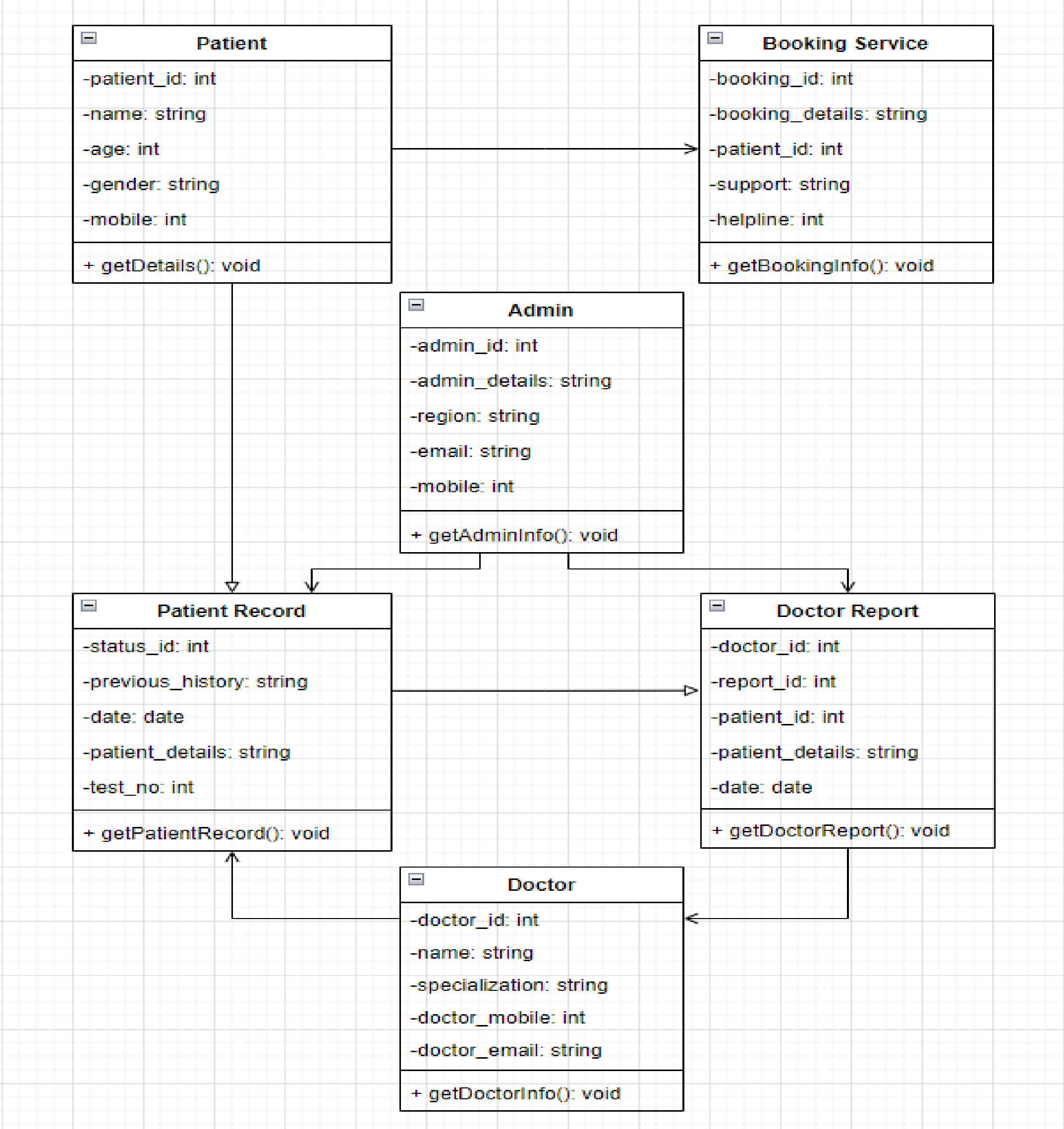


Figure 10: Class diagram of the project.

Chapter 4: Development Tools and Technologies

4.1 IDE

- Google Colab

4.2 Programming Language

- Python

4.3 Interface Design

The interface for this project was developed using machine learning AI in Google Collaboration.

4.4 Database

There is no specific database for this project.

4.5 Deploy and hosting

Open the web application or Android APK, then install the app on your device and run it.

Chapter 5: System Testing

5.1 Features

5.1.1 Predict Diabetes using Diabetes Data

5.1.2 Importing libraries

5.1.3 Read dataset

5.1.4 Train the model using the dataset

5.1.5 Check how many other missing (zero) values

5.1.6 Feedback

5.2 Strategies

5.2.1 Approach

- Manually tested.
- Provides accurate results.

5.2.2 Pass/Fail

- Pass successfully if the CSV file and machine learning model work properly.
- This project fails when a library or code error is detected.

5.2.4 Testing Schedule

Test Phase	Time	Owner
Test Plan Creation	1 week	Teertho Kamol
Test Specification Creation	2 week	Teertho Kamol
Test Specification Team Review	2 week	Teertho Kamol
Component Testing	2 week	Teertho Kamol
Integration Testing	1 week	Teertho Kamol
System Testing	1 week	Teertho Kamol

5.3 Test Cases

Test : 5.3.1	Test Name: CSV File Import
GlucoGuard Application	Subsystem: N/A

Designed by: Teertho Kamol	Design Date: 05-05-2024
Executed by: Teertho Kamol	Execution Date: 10-05-2024
Description: Import the library and CSV file to Google Colab.	
Prerequisites: N/A	

Test : 5.3.2	Test Name: Run All Libraries
GlucoGuard Application	Subsystem: N/A
Designed by: Teertho Kamol	Design Date: 11-05-2024
Executed by: Teertho Kamol	Execution Date: 18-05-2024
Description: Import the library and run all libraries.	
Prerequisites: N/A	

Test : 5.3.3	Test Name: Input The Data
GlucoGuard Application	Subsystem: N/A
Designed by: Teertho Kamol	Design Date: 19-05-2024
Executed by: Teertho Kamol	Execution Date: 25-05-2024
Description: Input valid data into the system.	
Prerequisites: N/A	

Chapter 6:User Manual

6.1 Import Drive on Colab

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Figure 11: Import Drive on Colab

6.2 Importing libraries

```
import pandas as pd
import matplotlib.pyplot as mplt
import numpy as np
from sklearn.impute import SimpleImputer
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
```

Figure 12: Importing libraries

6.3 Read dataset

```
[3] diabetes_data= pd.read_csv("/content/drive/MyDrive/Early Prediction of Diabetes Using Machine Learning/diabetes-dataset.csv")

[4] diabetes_data.shape

(2000, 9)

[5] diabetes_data.dtypes

Pregnancies                  int64
Glucose                      int64
BloodPressure                int64
SkinThickness                int64
Insulin                      int64
BMI                        float64
DiabetesPedigreeFunction   float64
Age                          int64
Outcome                      int64
dtype: object

[6] diabetes_data.isnull().values.any() #check the null value in dataset

False
```

```
diabetes_data.corr() #Correlation
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.120405	0.149672	-0.063375	-0.076600	0.019475	-0.025453	0.539457	0.224437
Glucose	0.120405	1.000000	0.138044	0.062368	0.320371	0.226864	0.123243	0.254496	0.458421
BloodPressure	0.149672	0.138044	1.000000	0.198800	0.087384	0.281545	0.051331	0.238375	0.075958
SkinThickness	-0.063375	0.062368	0.198800	1.000000	0.448859	0.393760	0.178299	-0.111034	0.076040
Insulin	-0.076600	0.320371	0.087384	0.448859	1.000000	0.223012	0.192719	-0.085879	0.120924
BMI	0.019475	0.226864	0.281545	0.393760	0.223012	1.000000	0.125719	0.038987	0.276726
DiabetesPedigreeFunction	-0.025453	0.123243	0.051331	0.178299	0.192719	0.125719	1.000000	0.026569	0.155459
Age	0.539457	0.254496	0.238375	-0.111034	-0.085879	0.038987	0.026569	1.000000	0.236509
Outcome	0.224437	0.458421	0.075958	0.076040	0.120924	0.276726	0.155459	0.236509	1.000000

```
[8] diabetes_data.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	3.703500	121.182500	69.145500	20.935000	80.254000	32.193000	0.470930	33.090500	0.342000
std	3.306063	32.068636	19.188315	16.103243	111.180534	8.149901	0.323553	11.786423	0.474498
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	63.500000	0.000000	0.000000	27.375000	0.244000	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	40.000000	32.300000	0.376000	29.000000	0.000000
75%	6.000000	141.000000	80.000000	32.000000	130.000000	36.800000	0.624000	40.000000	1.000000
max	17.000000	199.000000	122.000000	110.000000	744.000000	80.600000	2.420000	81.000000	1.000000

```
diabetes_data.groupby('Outcome').size()
```

```
Outcome
0    1316
1     684
dtype: int64
```

```
[10] diabetes_data.head(5)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	2	138	62	35	0	33.6	0.127	47	1
1	0	84	82	31	125	38.2	0.233	23	0
2	0	145	0	0	0	44.2	0.630	31	1
3	0	135	68	42	250	42.3	0.365	24	1
4	1	139	62	41	480	40.7	0.536	21	0

```
[11] diabetes_true_count = len(diabetes_data.loc[diabetes_data['Outcome'] == True])
     diabetes_false_count = len(diabetes_data.loc[diabetes_data['Outcome'] == False])
```

```
[13] (diabetes_true_count,diabetes_false_count)
```

```
(684, 1316)
```

Figure 13: Read the dataset

About Dataset

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether a patient has diabetes based on certain diagnostic measurements included in the dataset. Several constraints were placed on selecting these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage. 2 From the data set in the.csv file, we can find several variables, some of which are independent.
(several medical predictor variables) and only one target-dependent variable (outcome).

Information about dataset attributes:

- **Pregnancies:** To express the Number of pregnancies
- **Glucose:** To express the Glucose level in blood
- **BloodPressure:** To express the Blood pressure measurement
- **SkinThickness:** To express the thickness of the skin
- **Insulin:** To express the Insulin level in the blood
- **BMI:** To express the Body mass index
- **DiabetesPedigreeFunction:** To express the Diabetes percentage
- **Age:** To express the age
- **Outcome:** To express the final result 1 is Yes and 0 is No

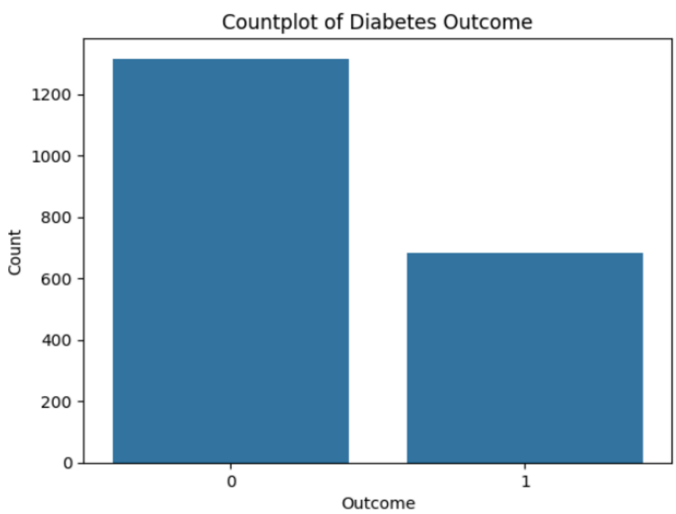


Figure 14: Countplot of diabetes outcome

6.4 Train the model using the dataset

```
[14] # Train-Test Split

     from sklearn.model_selection import train_test_split
     feature_columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']
     predicted_class = ['Outcome']
```

```
[15] X = diabetes_data[feature_columns].values
     y = diabetes_data[predicted_class].values


     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=10)
```

Figure 15: Train Train the model using the dataset

6.5 Check how many other missing (zero) values

```
[16] print("Total number of rows : {0}".format(len(diabetes_data)))
     print("Number of rows missing Pregnancies: {0}".format(len(diabetes_data.loc[diabetes_data['Pregnancies'] == 0])))
     print("Number of rows missing Glucose: {0}".format(len(diabetes_data.loc[diabetes_data['Glucose'] == 0])))
     print("Number of rows missing BloodPressure: {0}".format(len(diabetes_data.loc[diabetes_data['BloodPressure'] == 0])))
     print("Number of rows missing SkinThickness: {0}".format(len(diabetes_data.loc[diabetes_data['SkinThickness'] == 0])))
     print("Number of rows missing Insulin: {0}".format(len(diabetes_data.loc[diabetes_data['Insulin'] == 0])))
     print("Number of rows missing BMI: {0}".format(len(diabetes_data.loc[diabetes_data['BMI'] == 0])))
     print("Number of rows missing DiabetesPedigreeFunction: {0}".format(len(diabetes_data.loc[diabetes_data['DiabetesPedigreeFunction'] == 0])))
     print("Number of rows missing Age: {0}".format(len(diabetes_data.loc[diabetes_data['Age'] == 0])))

     Total number of rows : 2000
     Number of rows missing Pregnancies: 301
     Number of rows missing Glucose: 13
     Number of rows missing BloodPressure: 90
     Number of rows missing SkinThickness: 573
     Number of rows missing Insulin: 956
     Number of rows missing BMI: 28
     Number of rows missing DiabetesPedigreeFunction: 0
     Number of rows missing Age: 0
```

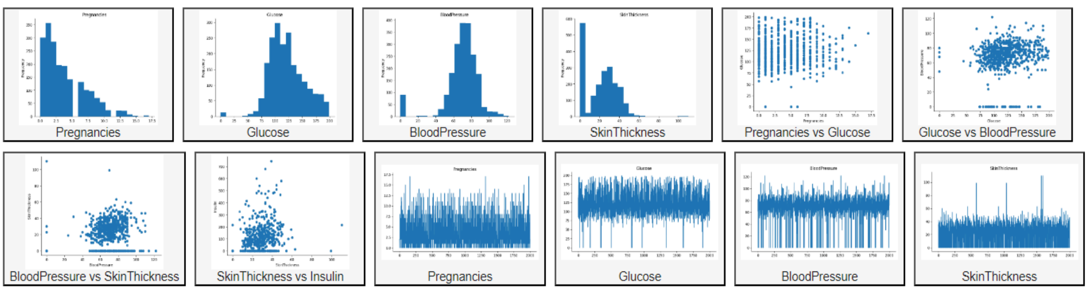
```
[17] from sklearn.impute import SimpleImputer
     fill_values =SimpleImputer(missing_values=0, strategy="mean")

     X_train = fill_values.fit_transform(X_train)
     X_test = fill_values.fit_transform(X_test)
```

```
[18] diabetes_data.head(5)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	2	138	62	35	0	33.6	0.127	47	1
1	0	84	82	31	125	38.2	0.233	23	0
2	0	145	0	0	0	44.2	0.630	31	1
3	0	135	68	42	250	42.3	0.365	24	1
4	1	139	62	41	480	40.7	0.536	21	0

Next steps: View recommended plots



```
[19] import seaborn as sns
     import matplotlib.pyplot as plt

     # Assuming diabetes_data is your DataFrame containing the 'Outcome' column
     sns.countplot(x='Outcome', data=diabetes_data)
     plt.title('Countplot of Diabetes Outcome')
     plt.xlabel('Outcome')
     plt.ylabel('Count')
     plt.show()
```

```
[20]
     sns.countplot(x='Pregnancies', hue='Outcome', data=diabetes_data, palette='colorblind', edgecolor=(0, 0, 0))
     plt.title('Countplot of Diabetes Outcome by Number of Pregnancies')
     plt.xlabel('Number of Pregnancies')
     plt.ylabel('Count')
     plt.show()
```

Figure 16: Check how many other missing (zero) values

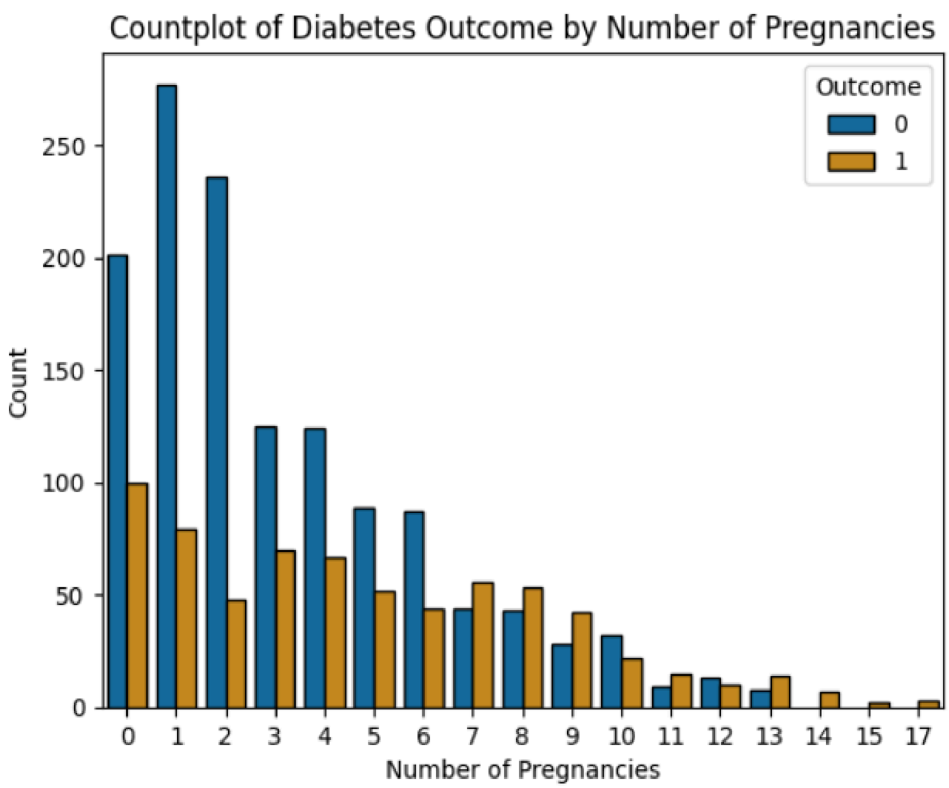
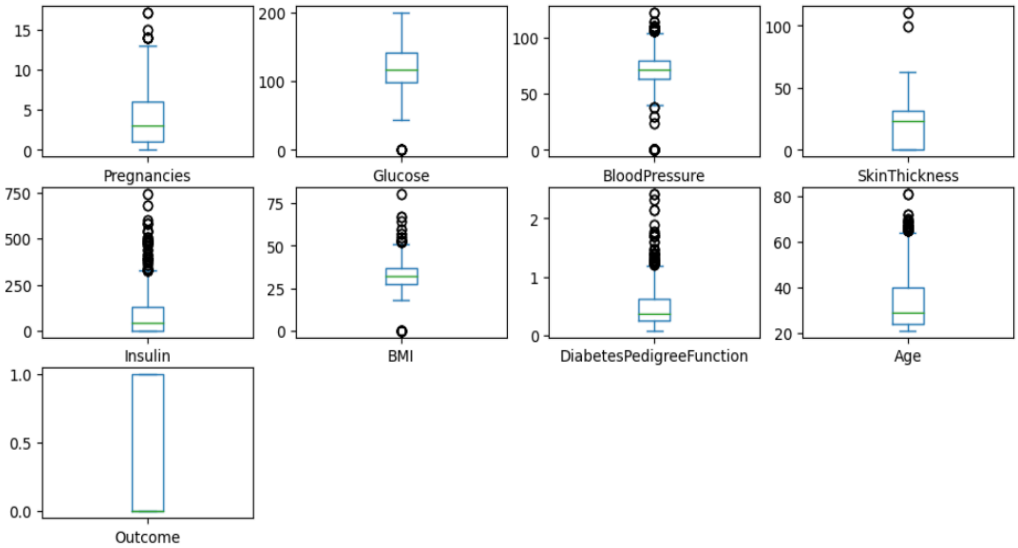


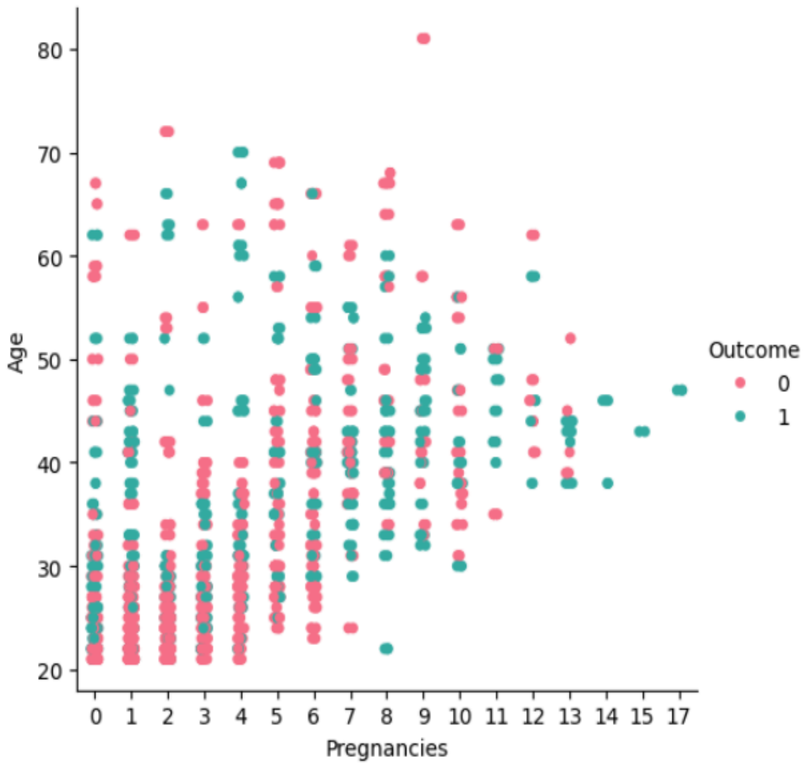
Figure 17: Countplot of diabetes outcome by number of pregnancies

```
[21] diabetes_data.plot(kind='box', subplots=True, figsize=(12,12), layout=(6,4))
     mplt.show()
```




```
[22] sns.catplot(data=diabetes_data, x='Pregnancies', y='Age', hue='Outcome', palette='husl')
```

<seaborn.axisgrid.FacetGrid at 0x7c6d1bf0ce50>



```
[23] sns.histplot(data=diabetes_data, x="Outcome")
```

```
<Axes: xlabel='Outcome', ylabel='Count'>
```

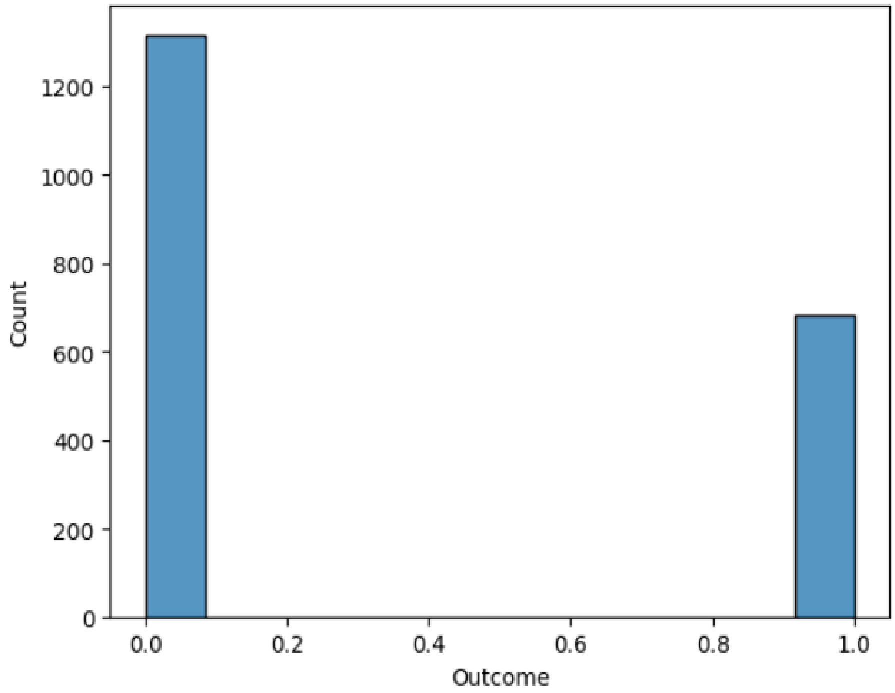


Figure 18: Diabetes data plot

```
[24] diabetes_data.hist()
```

```
array([[<Axes: title={'center': 'Pregnancies'}>,
        <Axes: title={'center': 'Glucose'}>,
        <Axes: title={'center': 'BloodPressure'}>],
       [<Axes: title={'center': 'SkinThickness'}>,
        <Axes: title={'center': 'Insulin'}>,
        <Axes: title={'center': 'BMI'}>],
       [<Axes: title={'center': 'DiabetesPedigreeFunction'}>,
        <Axes: title={'center': 'Age'}>,
        <Axes: title={'center': 'Outcome'}>]], dtype=object)
```

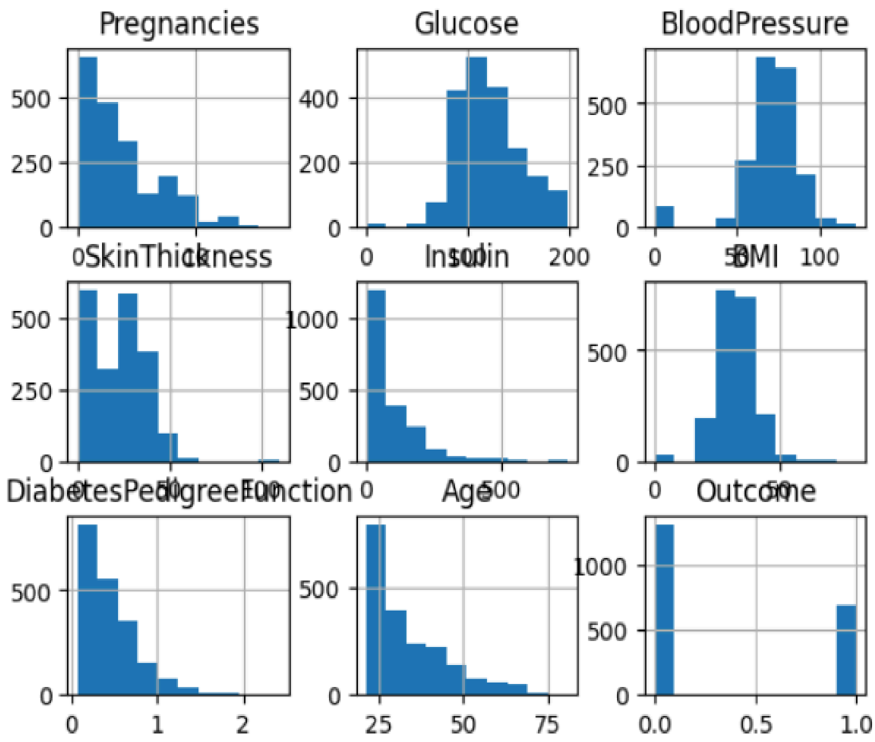


Figure 19: Diabetes data hist pie chart

Applying Algorithms:

- **KNeighborsClassifier:** Used for classification based on the k-nearest neighbors method.

```
[43] from sklearn.neighbors import KNeighborsClassifier
     knn = KNeighborsClassifier()
     knn.fit(X_train,y_train.ravel())
     knnpredict=knn.predict(X_test)
```

```
[44] knn.score(X_test,y_test)

     0.7933333333333333
```

- **Gaussian Process Classifier:** Predicts class probabilities by defining a distribution over functions and using observed data to update these probabilities for new inputs.

```
[45] from sklearn.gaussian_process import GaussianProcessClassifier
     gpc=GaussianProcessClassifier()
     gpc.fit(X_train,y_train.ravel())
     gpcpredict=gpc.predict(X_test)
```

```
[46] gpc.score(X_test,y_test)

     0.95
```

- **Gaussian Naive Bayes Classifier:** Used for its simplicity and effectiveness in handling continuous data by assuming that the features follow a normal distribution.

```
[47] from sklearn.naive_bayes import GaussianNB
     gnb=GaussianNB()
     gnb.fit(X_train,y_train.ravel())
     gnbpredict=gnb.predict(X_test)
```

```
gnb.score(X_test,y_test)

0.755
```

- **SVC (support-vector-machine-algorithm):** Used for classification and regression tasks because it finds the optimal hyperplane that maximizes the margin between different classes in the data.

```
[49] from sklearn.svm import SVC
     svc=SVC()
     svc.fit(X_train,y_train.ravel())
     svcpredict=svc.predict(X_test)
```

```
[50] svc.score(X_test,y_test)

     0.7916666666666666
```

● **RandomForestClassifier:** Used for its ability to handle large datasets with higher accuracy by averaging multiple decision trees to reduce overfitting and improve generalization.

```
from sklearn.ensemble import RandomForestClassifier
random_forest_model = RandomForestClassifier()

random_forest_model.fit(X_train, y_train.ravel())
#used to change a 2-dimensional array or a multi-dimensional array into a contiguous flattened array
```

```
▸ RandomForestClassifier
```

```
[52] predict_train_data = random_forest_model.predict(X_test)

     random_forest_model.score(X_test,y_test)
```

```
0.9683333333333334
```

● **DecisionTreeClassifier:** Used for its simplicity, interpretability, and ability to handle both numerical and categorical data for classification tasks.

```
[53] # Create Decision Tree classifer object
     dtc = DecisionTreeClassifier()

     # Train Decision Tree Classifer
     dtc = dtc.fit(X_train,y_train)

     #Predict the response for test dataset
     predictt = dtc.predict(X_test)
```

```
# Model Accuracy
dtc.score(X_test,y_test)
```

```
0.9283333333333333
```

- **Confusion Matrix:** Used to evaluate the performance of a classification algorithm by summarizing the counts of true positive, false positive, true negative, and false negative predictions.

```
[55] from sklearn.metrics import classification_report,confusion_matrix
     print(classification_report(y_test,knnpredict))
```

```
              precision    recall  f1-score   support

           0       0.84      0.85      0.85       398
           1       0.70      0.68      0.69       202

    accuracy                           0.79       600
   macro avg       0.77      0.76      0.77       600
weighted avg       0.79      0.79      0.79       600
```

```
[56] print(classification_report(y_test,gpcpredict))
```

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       398
           1       0.95      0.90      0.92       202

    accuracy                           0.95       600
   macro avg       0.95      0.94      0.94       600
weighted avg       0.95      0.95      0.95       600
```

```
[57] print(classification_report(y_test,gnbpredict))
```

```
              precision    recall  f1-score   support

           0       0.80      0.83      0.82       398
           1       0.65      0.60      0.62       202

    accuracy                           0.76       600
   macro avg       0.73      0.72      0.72       600
weighted avg       0.75      0.76      0.75       600
```

```
[58] print(classification_report(y_test,svcpredict))
```

```
              precision    recall  f1-score   support

           0       0.80      0.92      0.85       398
           1       0.77      0.54      0.64       202

    accuracy                           0.79       600
   macro avg       0.79      0.73      0.74       600
weighted avg       0.79      0.79      0.78       600
```

```
[59] print(classification_report(y_test,predict_train_data))
```

```
              precision    recall  f1-score   support

           0       0.97      0.98      0.98       398
           1       0.96      0.95      0.95       202

    accuracy                           0.97       600
   macro avg       0.97      0.96      0.96       600
weighted avg       0.97      0.97      0.97       600
```

```
[61] print(classification_report(y_test,predictt))
```

```
              precision    recall  f1-score   support

           0       0.92      0.97      0.95       398
           1       0.94      0.84      0.89       202

    accuracy                           0.93       600
   macro avg       0.93      0.91      0.92       600
weighted avg       0.93      0.93      0.93       600
```

- **Tree Visualization:** Tree visualization algorithms are used to represent hierarchical structures in a clear and understandable manner.

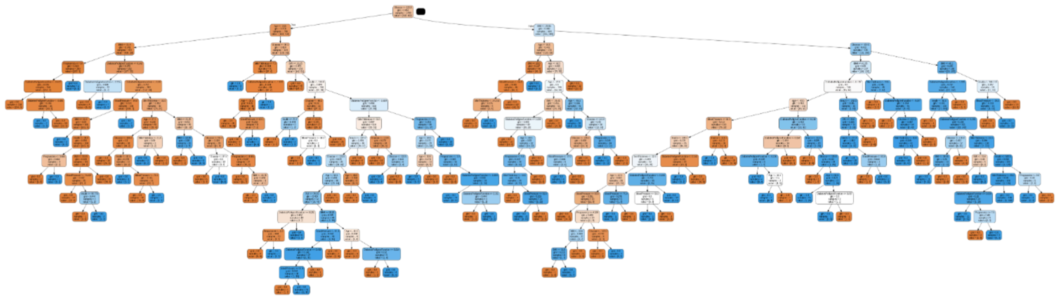
```
[62] import six #!pip install --upgrade scikit-learn==0.20.3
     import sys
     sys.modules['sklearn.externals.six'] = six
     from IPython.display import Image
     from sklearn.tree import export_graphviz
     import pydot

     features = list(diabetes_data.columns[:-1])
     features
```

```
['Pregnancies',
 'Glucose',
 'BloodPressure',
 'SkinThickness',
 'Insulin',
 'BMI',
 'DiabetesPedigreeFunction',
 'Age']
```

```
[63] from io import StringIO
dot_data = StringIO()
export_graphviz(dtc, out_file=dot_data,feature_names=features,filled=True,rounded=True)

graph = pydot.graph_from_dot_data(dot_data.getvalue())
Image(graph[0].create_png())
```



6.5 Patient Data Input in the System:

```
# predict user data

Pregnancies=int(input("Enter Number of Pregnancies: "))
Glucose=int(input("Enter Glucose: "))
BloodPressure=int(input("Enter Blood Pressure: "))
SkinThickness=int(input("Enter Skin Thickness: "))
Insulin=int(input("Enter Insulin: "))
BMI=float(input("Enter BMI: "))
DiabetesPedigreeFunction=float(input("Enter Diabetes Pedigree Function: "))
Age=int(input("Enter Age: "))

model= RandomForestClassifier()
model.fit(X,y.ravel())

predictions= model.predict([[Pregnancies,Glucose,BloodPressure,SkinThickness,Insulin,BMI,DiabetesPedigreeFunction,Age]])
if predictions==0:
    print("---------------------------")
    print("---------------------------")
    print('The person is not diabetic')
    print("---------------------------")
    print("---------------------------")
else:
    print("------------------------")
    print("------------------------")
    print('The person is diabetic')
    print("------------------------")
    print("------------------------")
```

```
Enter Number of Pregnancies: 3
Enter Glucose: 80
Enter Blood Pressure: 0
Enter Skin Thickness: 0
Enter Insulin: 0
Enter BMI: 0
Enter Diabetes Pedigree Function: 0.174
Enter Age: 22
```

6.6 The Result:

Person 1:

```
Enter Number of Pregnancies: 3
Enter Glucose: 80
Enter Blood Pressure: 0
Enter Skin Thickness: 0
Enter Insulin: 0
Enter BMI: 0
Enter Diabetes Pedigree Function: 0.174
Enter Age: 22
---------------------------
---------------------------
The person is not diabetic
---------------------------
---------------------------
```

Person 2:

```
Enter Number of Pregnancies: 7
Enter Glucose: 195
Enter Blood Pressure: 70
Enter Skin Thickness: 33
Enter Insulin: 145
Enter BMI: 25.1
Enter Diabetes Pedigree Function: 0.163
Enter Age: 55
------------------------
------------------------
The person is diabetic
------------------------
------------------------
```

Person 3:

```
Enter Number of Pregnancies: 0
Enter Glucose: 180
Enter Blood Pressure: 90
Enter Skin Thickness: 26
Enter Insulin: 90
Enter BMI: 36.5
Enter Diabetes Pedigree Function: 0.314
Enter Age: 35
------------------------
------------------------
The person is diabetic
------------------------
------------------------
```

Chapter 7: Conclusion

7.1 Github Link:

https://github.com/teertho-kamol/Early-Prediction-of-Diabetes-Using-Machine-Learning

7.2 Limitations

1. Data Privacy
2. Data Accuracy
3. Algorithm Accuracy
4. Device Compatibility
5. User Dependency
6. Cost
7. Technical Issues
8. Ethical Considerations
9. Regulatory Compliance

7.3 Future Scope

I will create a web application and an application for this endeavour in the future. That enables individuals to effortlessly access the application's advantages.

Appendix A

Reference

Link:

- https://www.niddk.nih.gov/about-niddk/research-areas/diabetes
- https://www.diabetesresearchclinicalpractice.com/article/S0168-8227(23)00807-0/fulltext
- https://drc.bmj.com/
- https://diabetesresearch.org/
- https://www.diabetes.org.uk/our-research

Plagiarism Test

201-35-3032

ORIGINALITY REPORT

19%	16%	4%	11%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

#	Source	Type	%
1	dspace.daffodilvarsity.edu.bd:8080	Internet Source	6%
2	Submitted to Arab Open University	Student Paper	2%
3	github.com	Internet Source	2%
4	Submitted to Open University of Cyprus	Student Paper	2%
5	Submitted to Deptford Township High School	Student Paper	1%
6	Submitted to Hong Kong Baptist University	Student Paper	1%
7	123dok.com	Internet Source	<1%
8	Submitted to University of Stirling	Student Paper	<1%
9	Submitted to Emirates Aviation College, Aerospace & Academic Studies	Student Paper	<1%

10	www.scribd.com Internet Source	<1 %
11	www.process.st Internet Source	<1 %
12	docplayer.net Internet Source	<1 %
13	Submitted to University of Essex Student Paper	<1 %
14	Submitted to University of Technology, Sydney Student Paper	<1 %
15	Submitted to IIT Delhi Student Paper	<1 %
16	Submitted to Rhodes University Student Paper	<1 %
17	www.coursehero.com Internet Source	<1 %
18	www.ijesr.org Internet Source	<1 %
19	www.ijraset.com Internet Source	<1 %
20	Submitted to HELP UNIVERSITY Student Paper	<1 %

21	Submitted to October University for Modern Sciences and Arts (MSA) Student Paper	<1 %
22	fastercapital.com Internet Source	<1 %
23	Submitted to Middlesex University Student Paper	<1 %
24	ouci.dntb.gov.ua Internet Source	<1 %
25	ripo.unibel.by Internet Source	<1 %
26	ro.co Internet Source	<1 %
27	Submitted to University of Northampton Student Paper	<1 %
28	Marcelo Sampaio de Alencar. "Communication, Management and Information Technology - International Conference on Communciation, Management and Information Technology (ICCMIT 2016, Cosenza, Italy, 26-29 April 2016)", CRC Press, 2019 Publication	<1 %