Tribhuvan University

Academia International College

Final Year Project Report

On

Diabetes Prediction System

[CSC 412]

Under the supervision of

"Mr. Bishwas Mathema"

Submitted by

Animesh Shakya (T.U. Exam Roll No. 29004/078)

Milan Bucha Magar (T.U. Exam Roll No. 29015/078)

Nitesh Rayamajhi (T.U. Exam Roll No. 29018/078)

Department of Computer Science and Information Technology

Academia International College

Institute of Science and Technology

Tribhuvan University

September, 2025

Tribhuvan University

Academia International College

Final Year Project Report

On

Diabetes Prediction System

[CSC 412]

A final year project submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Information Technology awarded by Tribhuvan University

Submitted by

Animesh Shakya (T.U. Exam Roll No. 29004/078)

Milan Bucha Magar (T.U. Exam Roll No. 29015/078)

Nitesh Rayamajhi (T.U. Exam Roll No. 29018/078)

Submitted to

Department of Computer Science and Information Technology

Academia International College

Institute of Science and Technology

Tribhuvan University

September, 2025

Tribhuvan University

Institute of Science and Technology

Academia International College

Department of Computer Science and Information Technology

Email: mail@academiacollege.edu.np

Supervisor's Recommendation

I hereby recommend that the project work report prepared under my supervision by Mr. Nitesh Rayamajhi (29018), Mr. Animesh Shakya (29004) and Mr. Milan Bahadur Bucha (29015) entitled "Diabetes Prediction System" be accepted as fulfilling in partial requirements for the degree of Bachelor of Science in Computer Science and Information Technology. In my best knowledge, this is an original work in Computer Science and Information Technology.

……..

Mr. Bishwas Mathema
Project Supervisor
HOD/Program Coordinator
Department of Computer Science and Information
Technology Academia International College
Gwarko, Lalitpur

Tribhuvan University

Department of Computer Science and Information Technology

Academia International College

Certificate of Approval

This is to certify that this project prepared by Mr. Nitesh Rayamajhi, Mr. Milan Bucha Magar and Mr. Animesh Shakya entitled "Diabetes Prediction System" in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Information Technology has been well studied. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

…….. Mr. Bishwas Mathema Project Supervisor Department of Computer Science and IT Academia International College	…….. Mr. Bishwas Mathema HOD/Program Coordinator Department of Computer Science and IT Academia International College
…….. Internal Examiner Academia International College	…….. External Examiner Central Department of CSIT Tribhuvan University

Acknowledgement

We owe my most profound appreciation to Academia International College for giving us a chance to work on this project as part of our syllabus.

Special thanks to our supervisor, Mr. Bishwas Mathema (HOD/Program Coordinator, Academia International College), for his consistent guidance, support, and feedback throughout the report's creation. We are generously obligated to him for providing this excellent opportunity to expand our knowledge. It helped us a lot to realize what we studied for.

We would like to express our sincere gratitude to all those individuals, families, friends, colleagues, and teachers for supporting and helping us a lot in finalizing this project within the limited time frame by providing valuable insights and feedback on the report.

Thanking You,

Animesh Shakya (T.U. Exam Roll No. 29004)
Milan Bucha Magar (T.U. Exam Roll No.29015)
Nitesh Rayamajhi (T.U. Exam Roll No. 29018)

Abstract

This system tackles the world diabetes epidemic by creating a predictive system from just seven days of an individual's health data. We extract inputs like glucose, blood pressure, activity, and sleep through a ETL pipeline and calculate key features using K-Best technique and trends with the assistance of DBSCAN. We then put the statistically most relevant features into our optimized Random Forest module. The system does far more than simply generate a generic risk percentage -- it produces a risk category and estimates a prospective timeframe for onset and identifies the leading contributing factors and generates individually tailored health recommendations. Built in Python and with a modular backend architecture, our system is in a ready form to be integrated into mobile or web applications using the power of REST APIs, laying the foundation towards future real-time monitoring capability. Our contribution is an efficient, understandable tool for proactive healthcare.

Keywords: ETL, DBSCAN, Random forest, K-Best, REST-API

Table of Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface
BMI	Body Mass Index
CSV	Comma-Separated Value
DBSCAN	Density Based Spatial Clustering of Applications with Noise
DRF	Django REST Framework
EHR	Electronic Health Records
ETL	Extract Transform Load
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IOT	Internet Of Things
MVC	Model View Controller
MVT	Model View Template
ORM	Object Realtional Model
PDF	Portable Document Format
REST	Representational State Transfer
SPA	Single Page Application

Chapter 1: Introduction

1.1. Introduction

Diabetes is a fast-growing worldwide health problem with millions of people at risk because of lifestyle and clinical factors. It is a chronic, metabolic disease characterized by elevated levels of blood glucose (or blood sugar), which leads over time to serious damage to the heart, blood vessels, eyes, kidneys and nerves. The most common is type 2 diabetes, usually in adults, which occurs when the body becomes resistant to insulin or doesn't make enough insulin. Detection at an early stage and timely intervention are very important in avoiding or preventing the development of diabetic complication at the onset. Conventional methods of diagnosis cannot support detection of gradual patterns of patient behavior and gradual changes in the body in the long term. The purpose of this project is filling that gap by creating a predictive system that utilizes the latest seven-day clinical data to determine the likelihood(risk) of a person developing diabetes.

The system employs machine learning in the form of the Random Forest algorithm to compute risk-based forecasts from predictive features. Another cornerstone of the project is a complete data preprocessing pipeline developed that facilitates efficient conversion of raw inputs into valuable features for model building. Feature importance analysis is also incorporated in the system that assists in discovering the most influential factors responsible for risk estimation and enhances the interpretation and personalization.

The result is presented in the form of a complete-stack web application with a React-based interactive frontend and a Python-driven backend that processes the data and provides prediction logic. The product is not just a useful execution of advanced computing methods but also helps in preventive healthcare through easy and intelligent risk assessment.

1.2. Problem Statement

Current diagnostic methods for diabetes are largely reactive, identifying the disease after its onset. This delays intervention and increases the risk of complications. Furthermore, individuals on medication often present normalized data, making it hard to assess their

true health status. There is a critical need for a proactive system that evaluates recent trends and multiple indicators to predict diabetes accurately.

1.3. Objectives

- To build a machine learning-based model that predicts diabetes using recent seven-day clinical data.
- To integrate statistical and trend-based features for enhanced accuracy.

1.4. Scope and Limitations

1.4.1. Scopes

- **Early Diabetes Risk Prediction:**

  The system is designed to predict the risk of developing Type 2 Diabetes using user data from the past 7 days, making it useful for early detection and proactive prevention.

- **Data-Driven Insights:**

  It supports preprocessing of health data such as blood glucose levels, blood pressure, age, BMI, gender and family history.

- **Automated Feature Selection:**

  Uses K- best statistic technique to select the most relevant features, reducing overfitting and improving model performance.

- **Model Training and Inference:**

  Implements Random Forest Classifier for risk prediction, DBSCAN for glucose pattern clustering, and StandardScaler for normalization. Trained models can be reused without retraining.

- **Modular and Scalable:**

  Codebase is structured for integration with Django (backend) and React (frontend), and can be extended for REST API deployment or smart device (IoT) integrations in the future.

1.4.2. Limitations

- **Lack of Real-Time Monitoring**

  The system processes batch data and does not support real-time sensor or wearable device data streaming in its current version.

- **Generalized Recommendations**

  Health recommendations are rule-based and derived from feature importance. They are not dynamically personalized by a clinical rules engine or medical ontology.

- **Static Trained Models**

  The system relies on pre-trained .joblib files. Any improvement in prediction accuracy requires retraining the model with updated datasets.

- **Binary Classification Only**

  The system predicts diabetes as either likely or unlikely (0 or 1). It does not distinguish between types of diabetes or severity levels.

- **Dataset Dependency**

  Accuracy and reliability depend heavily on the quality, size, and balance of the training dataset. Currently, it is trained on a static dataset and not validated across diverse populations.

- **No Direct Medical Integration**

  This system is an advisory tool and not integrated with electronic health records (EHR) or certified for clinical decision-making.

1.5. Methodology

Our project adopts the Incremental Development Model, which is especially suited for us academic and student-led software projects. In this model, the system is built in small, manageable components or "increments," each adding new functionality to the core system. Unlike models that rely on continuous end-user feedback (e.g., Agile or Prototype), the incremental model is ideal as we the development team ourselves are responsible for refining and reviewing the system.

We began by developing a minimal viable version of the system that accepts 1/7-day clinical input to predict diabetes risk using a Random Forest algorithm. As each

increment is completed and internally validated, new features such as data visualization, risk interpretation, medication-aware predictions, and report generation are gradually integrated into the system.

The Incremental Model works well in our context because:

- It allows us to build and test essential parts first, then add more advanced features.
- It does not depend on continuous user feedback, which aligns with our project's academic setting.
- Each new feature can be developed, tested, and added without disrupting the overall system.

The incremental model is a practical choice for our team project, enabling structured development while providing room for internal creativity and adjustment.

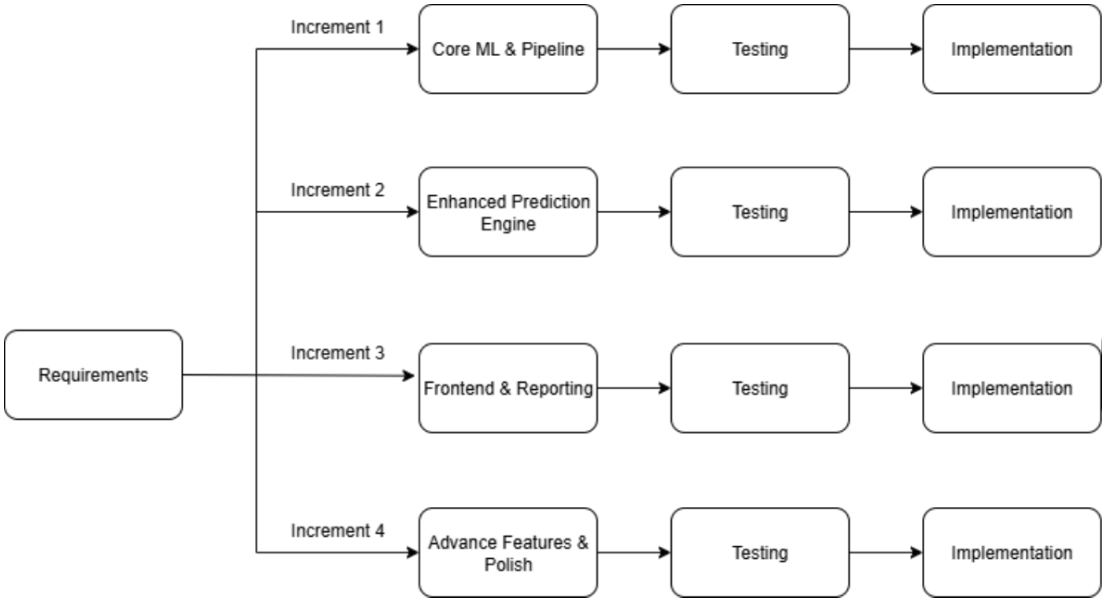

Figure 1: Incremental Methodology of the system

1.6. Report Organization

After successfully completing the project, a comprehensive project report has been prepared. The report begins with essential introductory sections, including the Title Page, Certificate Page, Acknowledgement, Abstract, Table of Contents, and lists of Abbreviations, Figures, and Tables.

The main body of the report is structured into six chapters, each focusing on a specific aspect of the project:

Chapter 1: Introduction

This chapter provides an overview of the project, covering key aspects such as the project introduction, problem statement, objectives, scope and limitations, and methodology.

Chapter 2: Background Study and Literature Review

Here, the project's background is discussed, along with a review of existing literature. This includes summaries of relevant projects, research papers, and articles that helped shape the project's direction.

Chapter 3: System Analysis

This section delves into system analysis, including requirements and feasibility studies. It defines the system's functional requirements using a use case diagram and presents a Gantt chart to visually illustrate the timeline and progress of various project tasks.

Chapter 4: System Design

This chapter explores the design phase in detail, covering the implementation process, model architecture, user interface, and system interactions. It also includes insights into the algorithms used in the project.

Chapter 5: Implementation and Testing

The focus here is on the implementation process and testing phases. It provides an overview of the tools and dependencies used to build the system and outlines the testing procedures undertaken to ensure functionality.

Chapter 6: Conclusion

The final chapter wraps up the project with a summary of key findings and conclusions highlighting potential areas for future improvements and enhancements.

Chapter 2: Background Study and Literature Review

2.1. Background Study

Diabetes mellitus is a chronic metabolic disorder that affects millions globally and has become a major public health concern. According to the World Health Organization (WHO), the number of people with diabetes has risen significantly in recent decades, with projections indicating continued growth due to lifestyle changes, poor diet, and sedentary habits. Early detection and prevention are critical in reducing the impact of diabetes and its associated complications such as cardiovascular diseases, kidney failure, and neuropathy.

Traditionally, diabetes diagnosis relies on clinical methods such as fasting blood glucose tests, HbA1c measurements, and oral glucose tolerance tests. However, these tests are typically conducted once symptoms become visible, by which time preventive options may be limited. Moreover, they do not offer predictive insights into how an individual's current lifestyle could influence their future risk.

The integration of machine learning (ML) in healthcare has opened new opportunities for data-driven preventive care. ML models can identify hidden patterns in patient health data, enabling early risk assessment and decision support. Various studies have demonstrated the feasibility of using classification algorithms like Decision Trees, Support Vector Machines, and Random Forests to predict diabetes risk based on factors such as age, BMI, glucose levels, physical activity, and family history.

Despite these advances, many existing prediction models are either too generalized or lack personalization based on recent lifestyle trends. In real-world settings, a person's health behavior over a short duration (such as a week) can be an important indicator of their metabolic stability or deterioration. A system that can learn from such short-term data and make predictive recommendations could support both individuals and healthcare professionals in taking timely action.

This project aims to bridge this gap by developing a machine learning-based diabetes risk prediction system that analyzes seven days of user input, including clinical and lifestyle metrics, to provide a personalized risk score, timeframe estimation, and targeted health recommendations. The model leverages statistical feature selection, clustering, and classification techniques to deliver interpretable and actionable insights.

2.2. Literature Review

For our project, we reviewed existing works related to diabetes prediction, lifestyle-based monitoring, and diet recommendation systems that utilize machine learning.

2.2.1 Diabetes Prediction Using Machine Learning

K. Vijayakumar proposed the Random Forest algorithm for the Prediction of diabetes to develop a system which can perform early prediction of diabetes for a patient with a higher accuracy by using Random Forest algorithm in machine learning technique. The proposed model gives the best results for diabetic prediction and the result showed that the prediction system is capable of predicting diabetes disease effectively, efficiently and most importantly, instantly.[1]

Nonso Nnameka presented predicting diabetes onset: an ensemble supervised learning approach they used five widely used classifiers are employed for the ensembles and a meta- classifier is used to aggregate their outputs. The results are presented and compared with similar studies that used the same dataset within the literature. It is shown that by using the proposed method, diabetes onset prediction can be done with higher accuracy.

Deeraj Shetty proposed diabetes disease prediction using data mining assemble Intelligent Diabetes Disease Prediction System that gives analysis of diabetes malady utilizing diabetes patient's database. In this system, they propose the use of algorithms like Bayesian and KNN (K-Nearest Neighbour) to apply diabetes patient's databases and analyze them by taking various attributes of diabetes for prediction of diabetes disease.

Muhammad Azeem Sarwar proposed a study on prediction of diabetes using machine learning algorithms in healthcare. They applied six different machine learning algorithms. Performance and accuracy of the applied algorithms is discussed and compared. Diabetes Prediction is becoming the area of interest for researchers to train the program to identify if the patient is diabetic or not by applying a proper classifier on the dataset. Based on previous research work, it has been observed that the classification process is not much improved. Hence a system is required as Diabetes Prediction is an important area in computers, to handle the issues identified based on previous research. [2]

The application of machine learning in diabetes prediction has been widely explored, with Random Forest (RF) consistently emerging as one of the most effective algorithms. Noviyanti and Alamsyah (2024) implemented RF on the Pima Indian Diabetes dataset,

achieving an accuracy of 87%, attributing this performance to its ability to handle nonlinear relationships and resist overfitting. This is corroborated by a large-scale community-based study in Guangzhou, which achieved a remarkable 91.24% accuracy and 97% AUC using RF on over 250,000 records. Comparative analyses, including a systematic review of 53 articles, affirm that ensemble methods like RF and XGBoost generally outperform single classifiers, with some studies, such as an ensemble of Naïve Bayes and SVM, reporting accuracies as high as 97.6%. Recent research continues to explore sophisticated approaches, such as enhanced deep neural networks and hybrid super ensemble models, highlighting a clear trend towards leveraging combined algorithmic strength to push the boundaries of prediction accuracy and robustness.[8]

Feature engineering and feature selection have been recognized as key components of maximizing model performance. Literature suggests choosing the most predictive features has a substantially positive impact on accuracy and decreases computational complexity. It has been shown that methods such as LASSO regression can be used very successfully to select clinically meaningful features highly related to pre-diabetes, e.g., age, triglycerides, BMI, and bad cholesterol. Also, the incorporation of lifestyle parameters—such as whether or not the patient smokes, physical activity level, and dietary habits—with clinical signs is a shift toward more complete models of diabetes risk and a greater understanding of the multi-dimensional manifestations of diabetes. Higher-order engineering concepts such as generating polynomial features, interaction terms (e.g., Glucose_BMI_Interaction), and ratio-based features (e.g., Insulin_Glucose_Ratio) assist with understanding complex non-linear physiological relationships. Other new approaches include utilizing density-based clustering (DBSCAN) to generate new features that detect non-typical patient profiles and hence enhance the ability of the model to describe diffuse manifestations of the disease.[6]

The field relies on a variety of diverse datasets, each with unique advantages and challenges. The Pima Indians Diabetes Database (PIDD) is frequently used despite its limitations in size and diversity, while larger surveys like the China Health and Nutrition Survey (CHNS) offer more comprehensive longitudinal data. The use of such datasets is fraught with data quality challenges, such as medically impossible zero values in features like Glucose and Insulin, which must be handled through sophisticated imputation techniques. The profound impact of preprocessing is underscored by studies showing that

model performance can dramatically improve with proper data cleaning, with one example showing Random Forest accuracy rising to 100% after preprocessing, emphasizing that this step is a crucial determinant of prediction success rather than a mere preliminary task.[5]

With increasingly complex models comes the greater need for interpretability and explainability for clinical use. The black-box nature of most deep learning and complex algorithms has prompted interest in Explainable AI (XAI) approaches. Methods like SHAP (Shapley Additive Explanations) and LIME are now deployed to shed light on the relative contribution of each feature individually to output predictions and build confidence and inform clinical implementation. For instance, a 2025 publication used SHAP analysis to uncover that age, BMI, and cholesterol were the strongest pre-diabetes risk factors with both quantitative ranking and directional impact available to inform cliniciansto guide their work. Beyond technical insights, visual methods like decision boundary plots and the creation of simple risk score cards have proven useful for translating the complex output of models into usable tools at the hands of community doctors and patients and closing the algorithm output and clinical utility gap.[4]

The emergence of telehealth and real-time monitoring technologies has profoundly increased data available for forecasting and control. Continuous Glucose Monitoring (CGM) system wearables, smart rings, and insulin pumps permit the retrieval of high-frequent real-world data and offer unprecedented information regarding dynamic glucose profiles and their drivers. Mobile apps function as unifying hubs within which these multifarious data streams are integrated and permit the user base to view measurements, access reports, and export data to healthcare professionals for remote review and prompt intervention. This unification of mobile health technologies and wearables and sophisticated ML algorithms holds unprecedented promise of personalized, anticipatory diabetes care pushing beyond the confines of conventional clinical environments toward everyday life with features of AI vocal recognition further facilitating use by all individuals.[3]

A promising application of ML lies in the development of personalized recommendation systems. Research has led to systems that generate tailored health and nutrition advice based on individual factors like age, gender, daily meals, and exercise intensity. These

systems use algorithms ranging from Nearest Neighbor to collaborative filtering to create customized diet plans and provide users with a comprehensive overview of their nutritional status via visualizations. The integration of these recommendation systems with predictive models creates a powerful synergy, shifting the focus from passive prediction to active intervention. This allows for tailored guidance on diet, exercise, and sleep, which is crucial as studies show timely lifestyle interventions can reduce the progression to T2DM by 40-70%, potentially improving adherence and effectiveness compared to generic advice. Despite remarkable progress, the area still has limitations and knowledge gaps. Numerous works remain based on rather small or biased datasets, which poses generalizability concerns. Also, numerous models are crafted within controlled environments without proper validation within real-world clinical scenarios, calling their real-world applicability into question. Heavy dependence on conventional clinical metrics is still maintained with frequent neglect of crucial behavioral and environmental determinants and limited attention paid to temporal phenomena of disease evolution. Future work must hence concentrate on making use of larger and more diverse datasets, exploiting more complex feature engineering that accounts for temporal phenomena, improving interpretation of models, and exploiting new technologies such as non-invasive monitors to transform data acquisition and boost prediction accuracy with real-world impact.[7]

Most existing systems focus either solely on clinical data or require expensive continuous monitoring. Many lack actionable lifestyle guidance and do not incorporate user-specific insights. Our system aims to fill this gap by offering a lightweight, user-friendly platform that combines both data types and provides personalized suggestions based on feature analysis.

Chapter 3: System Analysis

3.1. Requirement Analysis

Before beginning the development of the Diabetes Prediction and Monitoring System, a thorough system analysis was performed to ensure that the final product meets both the technical and user-centric goals. This analysis included identifying functional and non-functional requirements necessary for efficient system performance, usability, and reliability.

The requirements for the system are divided into Functional and Non-Functional categories.

3.1.1. Functional Requirements

The following core functionalities are required for the system:

- **User Registration and Authentication**

  Secure login and registration functionality to manage individual user profiles.

- **Daily Health Data Entry**

  Interface to allow users to input 7-day data including fasting/postmeal glucose, blood pressure, diet, sleep, and other attributes.

- **Data Preprocessing and Feature Engineering**

  Automatically process the input data using rolling averages, trend analysis, and clustering methods.

- **Diabetes Risk Prediction**

  Use a machine learning model (Random Forest) to predict the risk percentage of developing diabetes.

- **Key Factor Identification**

  Show the top health factors contributing to the risk to improve transparency and interpretability.

- **Data Storage and Management**

  Store user records and predictions in a secure backend database for future reference.

Figure 2: Use Case Diagram

The use case diagram illustrates the interaction between the User and the Diabetes Prediction System. The user engages with the system through a set of core functionalities designed to monitor and predict diabetes risk based on health data. Each oval in the diagram represents a specific function or use case that the user can perform or trigger.

The User can:

- **Submit Health Data**: This includes providing daily inputs such as glucose levels, blood pressure, physical activity, BMI, sleep hours, and carbohydrate intake over a 7-day period. This process includes the validation of inputs, ensuring that all necessary data is correctly formatted and complete before being processed.
- **Predict Diabetes Risk**: After data submission, the system processes the

information using a trained machine learning model to predict the user's likelihood of developing diabetes. The prediction also considers trends, variability, and clusters in the data.

- **View Risk Results**: The user can view the risk percentage, risk category (e.g., Low, Moderate, High), estimated timeframe for developing diabetes, and contributing factors. This view includes the generation of visualizations such as trend graphs and bar graphs showing influential features and risk progression.
- **Download Report**: Optionally, the user can download a detailed report summarizing their prediction, risk factors, timeframe estimation, and personalized health recommendations. This functionality extends the "View Risk Results" use case, meaning it is only accessible after viewing the prediction results.

3.1.2. Non-Functional Requirements

The system also adheres to the following non-functional requirements:

- **Accessibility and Responsiveness:** The system is accessible across devices (desktop and mobile) and responsive to different screen sizes.
- **User-Friendly Interface:** The UI is made to be clean, intuitive, and designed for ease of use, even for non-technical users.
- **System Performance and Scalability:** The model is able to respond within a reasonable time frame (~1–2 seconds per prediction) and be scalable to support a growing number of users.
- **Model Accuracy and Interpretability:** The prediction engine maintains functional accuracy and offer explainable insights (e.g., key contributing factors) rather than black-box results.
- **Security and Privacy:** All user data is handled securely, complying with data protection standards, ensuring encrypted storage and authenticated access.

3.2. Feasibility Analysis

Feasibility analysis is conducted to determine whether a project is technically, economically, operationally, and schedule-wise feasible to guarantee successful project completion.

3.2.1. Technical Feasibility

The project is technically feasible as it leverages reliable and well-supported open-source tools and publicly available health datasets.

- **Frontend Development**:

  Frontend development is handled using React, which is ideal for building responsive and interactive user interfaces. Data visualization tools like Chart.js and Recharts can be easily integrated into React, ensuring an engaging and informative UI for users.

- **Backend Development**:

  Backend development uses Python with scikit-learn, a widely adopted machine learning library. For data preprocessing, ETL tool ensures smooth transformation and loading of structured datasets, while the Random Forest algorithm provides accurate classification for diabetes prediction.

- **Training Data**:

  Training data such as the Pima Indian Datasets are publicly available, making data collection feasible.

- **Machine Learning Tasks**:

  The machine learning tasks involved (such as binary classification, feature importance, and risk prediction) are well-suited to the academic environment and achievable within a semester-long project timeline.

3.2.2. Operational Feasibility

Operational feasibility refers to the degree to which the proposed system is supported by the existing operational procedures and will function effectively within the current or anticipated environment.

In the case of this Diabetes Prediction System, Operational feasibility is very feasible

based on considerations of user accessibility, integration of workflows, and maintainability.

The system has a minimal and user-friendly interface with clearly labeled data entry fields and validation checks, so it can be used by non-technical individuals. It needs only seven days of health measurements and promotes frequent use with low user burden. It has output of actionable results such as risk levels, time horizon estimation, and drivers that aid usability and acceptability. Modular architecture supports maintainability and future upgrades like retraining of models or incorporation with wearable technology. It is web-based and platform-independent and can run on any device with a web browser. Optional offline functionality of data preparation enhances operational portability further too.

3.2.3. Economic Feasibility

The project is economically feasible as it incurs no direct financial cost to the development team. This is made possible through the use of:

- **Open-source Technologies**:

  All tools and frameworks used, such as React, Python, scikit-learn, are freely available and well-documented. There are no licensing or subscription fees required.

- **Free Datasets**:

  We utilize publicly available datasets such as PIMA Indian dataset from the UCI Machine Learning Repository, avoiding the need for any paid or proprietary data sources.

- **Local Development Environment**:

  The project can be developed and tested entirely using personal computers without the need for paid cloud services. If deployment is needed, free-tier services (e.g., GitHub Pages, Vercel, or Render for hosting, or Google Colab for ML testing) are sufficient for demonstration.

- **No Additional Hardware/Software Investment**:

  All development is done using existing resources available to the students. No new purchases are necessary for software licenses, hardware upgrades, or paid development platforms.

Though developed at absolutely no cost by taking advantage of open-source technologies and available resources, its commercial equivalent would entail a heavy investment. Below is the table of the approximate market cost if the project was developed by a professional team showing the cost-effectiveness of our work.

Table 3.1: Estimated Project Cost

Cost Component	**Assumption / Calculation**	**Estimated Cost (NPR)**	**Justification**
ML Engineer (1)	2 months @ NPR 30,000/month	60,000	Development of data pipeline, feature engineering, and machine learning model.
Backend Developer (1)	2 months @ NPR 35,000/month	70,000	Building the FastAPI backend, database design, and API integration.
Frontend Developer (1)	2 months @ NPR 25,000/month	50,000	Development of the React.js user interface and data visualization components.
Miscellaneous & Contingency		10,000	Buffer for unforeseen expenses.
Total Estimated Cost		190,000	

3.2.4. Schedule Feasibility

The project was completed within the estimated timeframe, accounting for all necessary tasks including game design, asset creation, programming, and playtesting. With a well- planned timeline, the project was divided into manageable milestones, ensuring progress was made efficiently and within the given schedule.

The following Gantt chart depicts the schedule established for the development of the system.

Task	W1-2	W3-4	W5-6	W7	W8
Documentation					
Data gathering & preprocessing					
Feature engineering & statistical analysis					
Model training & testing					
Frontend integration					
Final testing					

Figure 3: Gantt Chart

3.3. Analysis (Object-Oriented)

3.3.1. Object Modeling using Class and Object Diagrams

This class diagram models a simple diabetes prediction system where a User represents someone using the system, identified by a user ID and name. Each user can submit their health-related information using the submitData() method. The submitted data is stored in the HealthData class, which includes key health indicators such as glucose level, BMI,

and age. A user can have multiple health data entries, as shown by the one-to-many relationship.

This health data is then passed into the PredictionModel class, which uses a trained RandomForest model to evaluate the risk of developing diabetes. It provides two core functions: predictRisk() to assess the likelihood of diabetes and getFeatureImportance() to determine which health features most influenced the prediction.

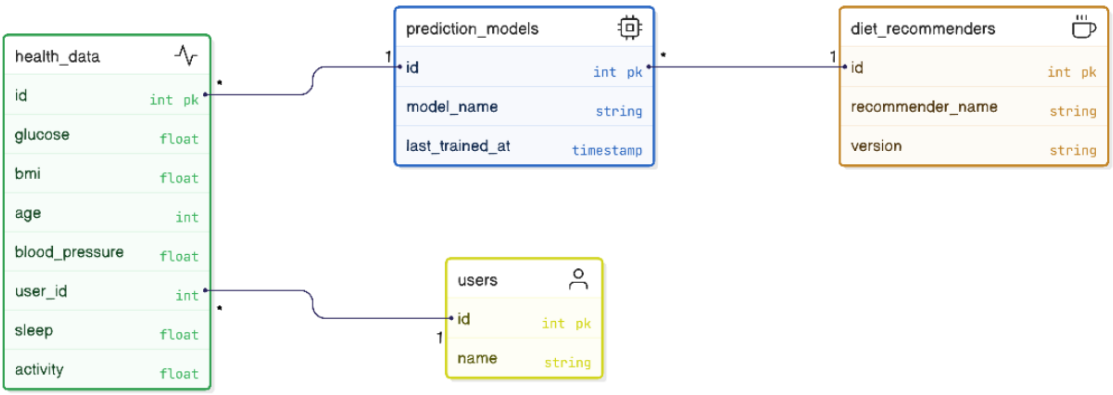

Figure 4: Class Diagram

Based on the results from the prediction model, the DietRecommender class generates personalized dietary suggestions through its generatePlan() method, aiming to help the user manage or reduce their diabetes risk.

Overall, the diagram illustrates the flow of data from user input to health analysis and finally to diet recommendations, with each class playing a distinct role in the process.

3.3.2. Dynamic modelling using state and sequence diagram

This sequence diagram illustrates how our diabetes prediction system operates across various components. A user begins by submitting 7-day health data (e.g., glucose, BMI) through the React frontend. The frontend sends this data to the Django backend via an API call. The backend forwards the data for preprocessing (such as cleaning and transforming), and once processed, it's returned to the backend.

Next, the backend sends the cleaned data to the ML model (using Random Forest) to request a risk prediction. The ML model returns the diabetes risk percentage along with key influencing features. Optionally, the backend stores the prediction result in the database. The backend then generates diet recommendations and sends both the prediction and recommendations back to the frontend, which finally displays the results

like charts and risk scores—to the user.

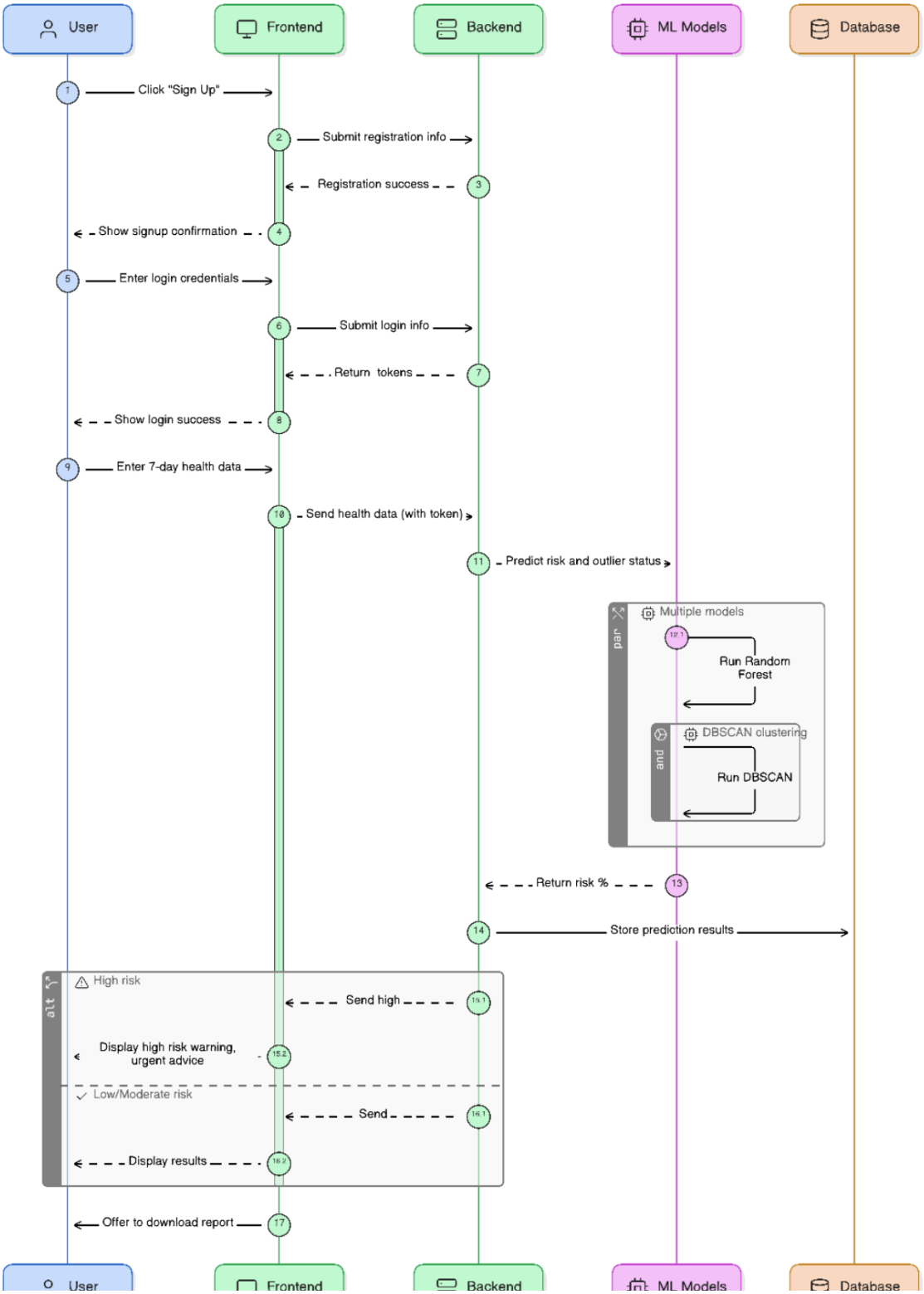

Figure 5: Sequence Diagram

3.3.3. Process modelling using activity diagram

An activity diagram is a UML behavioral diagram that visualizes the step-by-step workflow of a system, including actions, decisions, and parallel processes. It helps in projects by providing a clear, visual representation of how different components like users and systems interact, ensuring smooth functionality and early detection of potential issues.

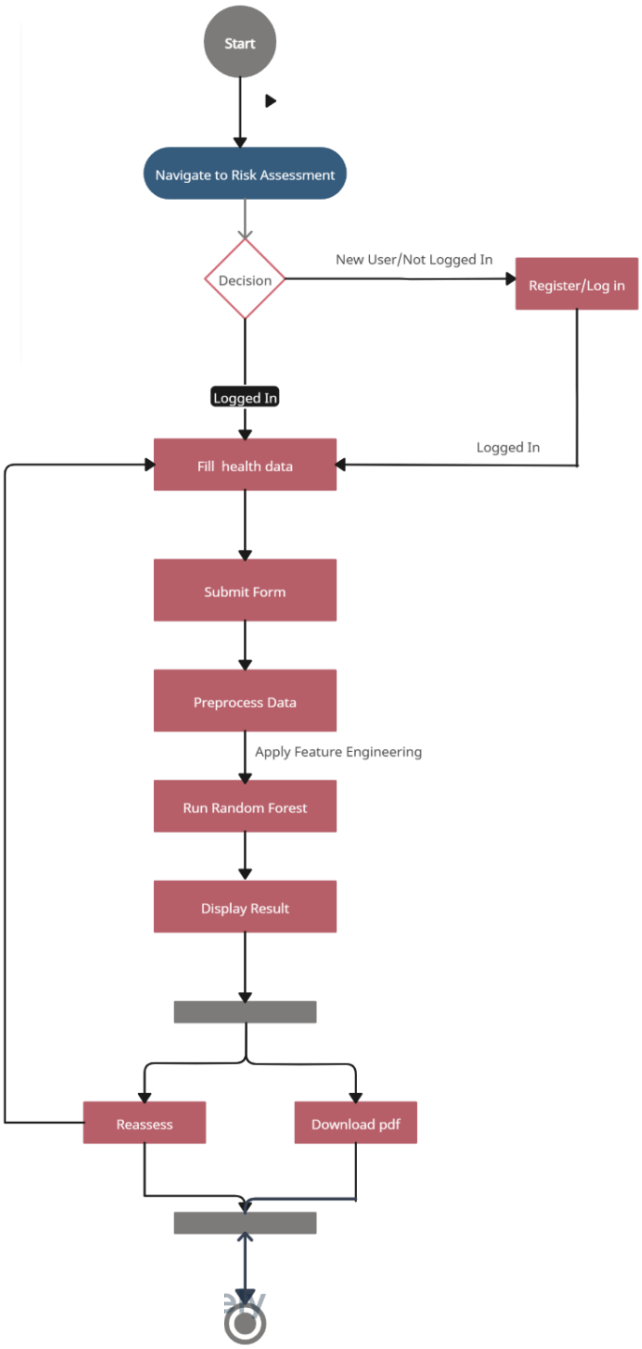

Figure 6: Activity Diagram

Chapter 4: System Design

4.1. Design

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. It focuses on planning the structure and behavior of a system, ensuring that all parts work together efficiently to achieve the intended objectives.

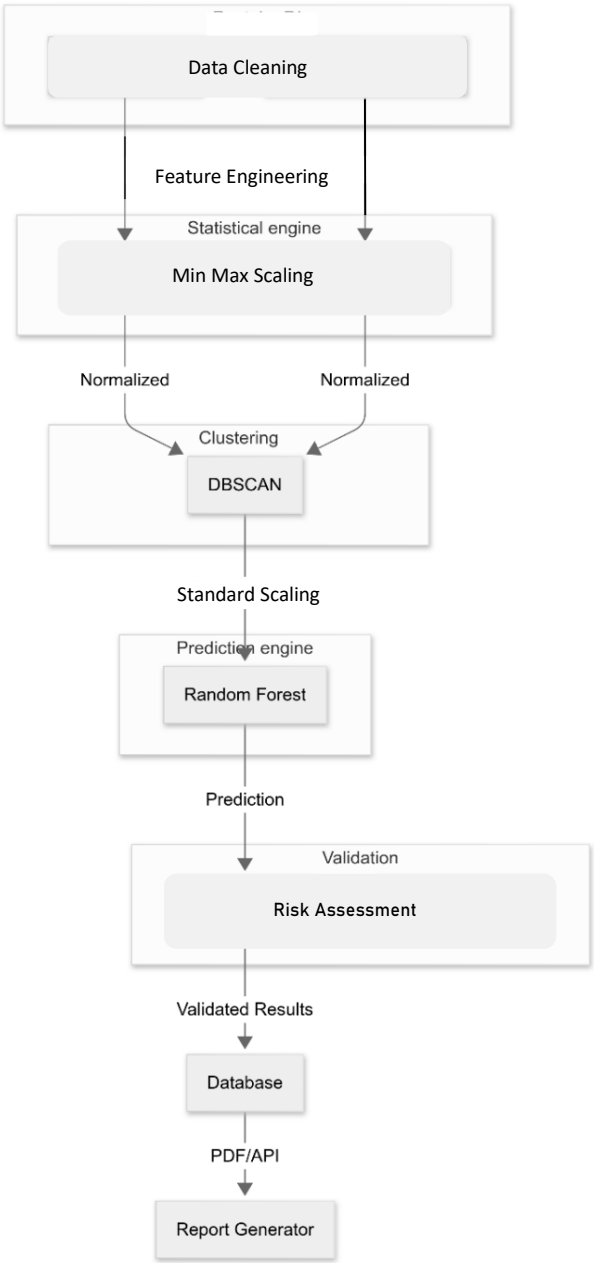

Figure 7: System Flow Diagram

4.2. System Architecture

Diabetes Prediction System is developed as a decentralized web application based on a client-server architecture. This architecture is divided into four individual layers Client, Application Server, Machine Learning Service, and Data—and communicate over a network. This clear-cut decomposition of responsibilities enhances scalability and maintainability and allows individual development and deployment of each module independently.

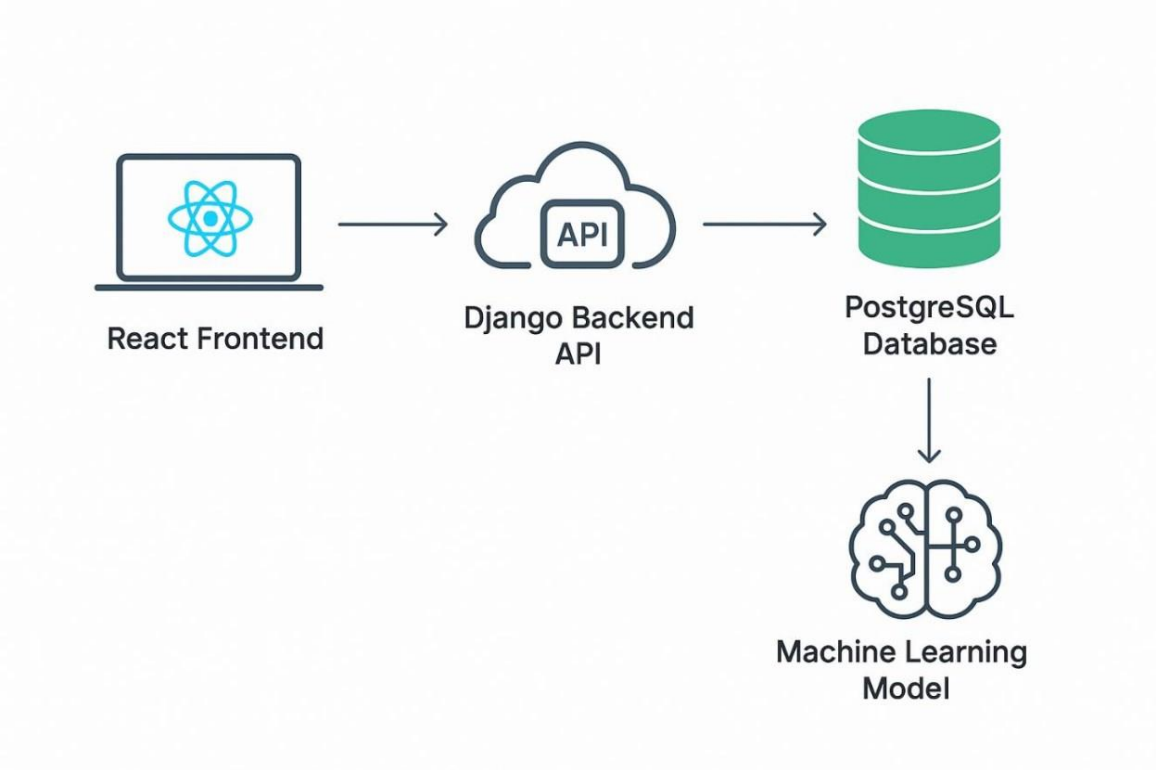

Figure 8: System Architecture

- **Client Layer**

This is the system's user interface and is charged with displaying information and accepting user input. It is developed as a React.js single-page application (SPA) and runs within contemporary web and mobile browsers. This technology choice guarantees a dynamic and interactive user interface where changes of the content occur effortlessly using full page refreshes. This Client Layer communicates with the backend using only HTTP requests directed at a RESTful API and sending and receiving all data using the JSON format.

- **Application Server Layer**

As the base business logic layer, the Application Server Layer handles every client request, enforces agreed business rules, and orchestrates system operation. It is developed using Django and the Django REST Framework (DRF), and is generally deployed on a WSGI server such as Gunicorn. It is mostly concerned with user authorisation and authentication, request validation and routing, and management of the full prediction flow. It is equally charged with all interactions with the Data Layer using the Django ORM and is the central go-between of the application.

- **Machine Learning Service Layer**

This is where all data science and predictive analytics functionality resides, crafted as a distinct, callable service for utmost modularity. Components are pre-trained Random Forest and DBSCAN models and their corresponding preprocessing pipelines developed using Scikit-learn and Joblib. These pre-serialized models (files ending in .pkl or .joblib) are currently read by Application Server directly within our implementation but can evolve toward a complete independent microservice with its own API. 4.1.4 Data Layer Data Layer secures permanent storage of all structured system data using PostgreSQL, a powerful relational database system. It safely stores user account data and credentials, user-submitted chronic health data and results of all past predictions and recommended items. Access to the databases is made and managed exclusively by the Application Server Layer using only the Django ORM, maintaining data integrity and security throughout the system.

4.3. Application Architecture

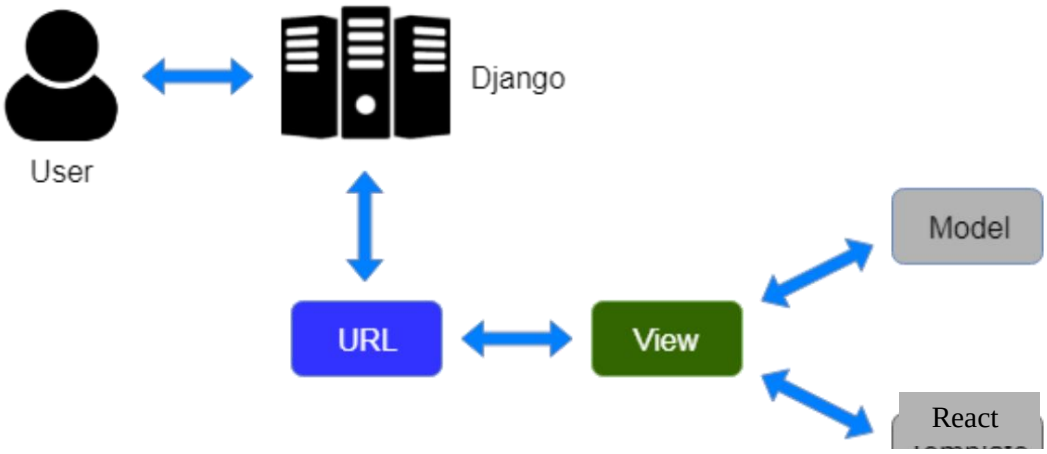

Figure 9: MVT Architecture (React instead of Template)

The system utilizes a Model-View-Template architecture that has been modified, a fundamental pattern of Django, but with a significant change: it substitutes a Template layer with a completely decoupled React.js client side, making it a Model-View-Client architecture at the full-stack level. This contemporary approach takes advantage of Django for data and logic (Model-View) and passes all of the UI rendering and user interaction duties entirely to the dynamic React client.

- **Model (Django)**

  The Model component remains firmly within the Django backend, implemented using Django's Object-Relational Mapper. This layer is responsible for the fundamental structure and integrity of all application data. Its core responsibilities include defining the schema and relationships for all data entities, such as the User model, the HealthData model, and the PredictionResult model. Furthermore, it serves as a complete abstraction layer for the PostgreSQL database, handling all create, read, update, and delete operations.

- **View (Django REST Framework)**

  In this adapted architecture, the role of the Django View component has evolved significantly. It no longer renders HTML templates but instead acts purely as a Controller (in classic MVC terminology) that processes requests and returns structured JSON responses. Built using the Django REST Framework, this layer is responsible for providing a complete set of RESTful API endpoints.

- **Client (React.js) - Replacing the Template**

  The Template layer of traditional MVT is entirely replaced by a sophisticated, self-contained React.js client that runs in the user's browser. This frontend application is responsible for the entire user experience. It dynamically renders all HTML content based on the application's state, creating a highly interactive and responsive single-page application that requires no full page reloads. It handles all user interactions, including form input, button clicks, and navigation. The React app manages its own client-side logic and state—such as form data, UI state, and prediction results and communicates with the backend exclusively by making HTTP requests to the Django REST API to fetch or send data.

4.4. Algorithm Details

4.4.1 Random Forest: The Core Predictor

The system employs a Random Forest Classifier as its primary engine for predicting diabetes risk. Random Forest is an ensemble learning method that constructs a multitude of decision trees and combines their predictions to achieve higher accuracy and stability than a single tree. Each tree is trained on a random bootstrap sample of the dataset (sampling with replacement), and at each split, only a random subset of features is considered. This incorporation of randomness encourages diversity among the trees, reducing overfitting and improving generalization.

$$Gini(S) = 1 - \sum_{i=1}^{C} p_i^2$$

Within each tree, splits are determined using Gini Impurity, a measure of node purity. Gini Impurity quantifies the likelihood of misclassifying a randomly chosen sample if it were labeled according to the distribution of classes in the node. It is mathematically defined as:

where:

- C is the number of classes (in this case, 2: diabetic and non-diabetic),
- p_i is the proportion of samples in class i at the node S.

A Gini Impurity of 0 indicates a perfectly pure node (all samples belong to one class), whereas a higher value indicates greater class mixing.

The root node is chosen by evaluating all candidate features and selecting the one that minimizes the weighted Gini Impurity of its child nodes after a split. Formally, for a candidate split that divides node S into child nodes S_L and S_R:

$$Gini_{split} = \frac{|S_L|}{|S|} \cdot Gini(S_L) + \frac{|S_R|}{|S|} \cdot Gini(S_R)$$

The feature with the lowest $Gini_{split}$ is selected as the root node because it best separates

diabetic from non-diabetic patients in the training dataset. Subsequent nodes are selected similarly, recursively minimizing impurity until stopping criteria—such as maximum depth or minimum samples per leaf—are reached.

Once trained, each tree independently produces a class prediction. The Random Forest c combines these predictions via majority voting, and probability estimates for a patient are computed as the fraction of trees predicting the positive class. For example, if 140 out of 200 trees classify a patient as "At Risk," the final probability is 70%.

The model is fine-tuned with hyperparameters including n_estimators=200 (number of trees), max_depth=10, min_samples_split=5, min_samples_leaf=2, and max_features = 'sqrt' to balance complexity with generalization. class_weight='balanced' is used to address the class imbalance between diabetic and non-diabetic samples.

A key advantage of Random Forest is its ability to compute feature importance by assessing the reduction in Gini Impurity contributed by each variable across all trees. In this system, glucose, BMI, age, and blood pressure consistently emerge as the most influential features, aiding patient-specific explanations and highlighting dominant risk factors in reports.

4.4.2 DBSCAN: The Outlier Detector

Alongside Random Forest, the system employs DBSCAN to detect atypical patient profiles. DBSCAN is a clustering algorithm that groups data points based on their density. Patients are clustered together if they are closely packed within a defined distance threshold (eps), while patients with fewer than a minimum number of neighbors are labeled as outliers. These outliers often correspond to unusual clinical cases that deviate from common patient patterns, such as a combination of extremely high glucose levels with normal BMI, which may confuse traditional classifiers.

In the system, DBSCAN plays a supportive but critical role by flagging suspicious cases that require further review. Such cases are particularly important in healthcare settings where automated models may misclassify rare but clinically significant patterns. To ensure meaningful clustering, the system preprocesses data using MinMax Scaler before applying DBSCAN, as the algorithm relies on Euclidean distance and is sensitive to differences in feature scales. By isolating anomalies, DBSCAN adds an extra safety layer, ensuring that flagged cases are highlighted for clinician oversight rather than being

overlooked by the core prediction engine. DBSCAN is performed after feature engineering and before the random forest prediction.

4.4.3 Averages: The Temporal Smoother

Applied to time-series inputs (e.g., daily glucose logs), 7-day rolling averages smooth noisy measurements. This preprocessing step ensures DBSCAN receive stable trends rather than erratic daily values. For example, a single high glucose spike won't skew clustering or statistical tests—only sustained elevations will.

4.4.4 StandardScaler

The random forest classifier users the standard scaler to produce the required prediction. Hence before the random forest, the data is scaled using standard scaler and the data is normalized using min-max scaler before the DBSCAN clustering.

Chapter 5: Implementation and Testing

5.1. Implementation

The Diabetes Prediction System was built step by step, starting with the data pipeline to collect and clean patient records, including vital signs like glucose levels and genetic factors. The data was processed to calculate rolling averages and normalized to ensure consistency. Key features were then selected using statistical tests to focus on the most relevant predictors.

At the core of the system, machine learning models were implemented—Random Forest for predicting diabetes risk, DBSCAN for detecting unusual patient patterns, and Naive Bayes to validate results. These models were rigorously tested using cross-validation to ensure accuracy and reliability.

The backend, developed with FastAPI, handles prediction requests and stores data in PostgreSQL, while the frontend, built with React, provides an intuitive interface for users to input data and view their risk assessments. Detailed PDF reports explain the results, highlighting key factors like glucose levels or BMI.

Scikit-learn powered the machine learning. The system was refined through continuous testing, resulting in a robust and user-friendly platform for diabetes risk prediction.

5.1.1. Tools Used

- **FastAPI**: Primary backend framework for building the REST API
- **React.js**: Frontend library for the user interface
- **PostgreSQL**: Database for storing patient records and predictions
- **Scikit-learn**: Machine learning library (Random Forest, DBSCAN, StandardScaler)
- **Pandas/NumPy**: Data processing and feature engineering
- **Git**: Version control system
- **GitHub**: Repository hosting and collaboration
- **VS Code**: Code editor with Python/React extensions
- **Postman**: API testing and documentation

5.1.2. Implementation Details of Modules

Some of the modules included in the project are:

5.1.2.1 Data Ingestion Module

We used ETL workflow that ingests raw user data from CSV files. This module handles cleaning, missing value imputation, and calculates critical 7-day rolling averages for time-series data like glucose levels, ensuring a stable dataset for analysis.

5.1.2.2 Feature Engineering Module

The testing process for the Diabetes Prediction System was carried out systematically across several stages. At first the raw diabetes.csv file was successfully loaded, producing a data frame with 768 samples and 8 features. Then after confirmed the generation of 15 engineered features, which were correctly added to the dataset. Thirdly Min Max scaling was applied to ensure all features were scaled within the range [0,1], while in next case standard scaling was performed, resulting in all features having a mean of 0 and a standard deviation of 1, suitable for Random Forest. DBSCAN clustering was applied, creating cluster features labeled Cluster_0, Cluster_1, Cluster_2, and Cluster_Outlier. Finally, feature selection was validated using SelectKBest successfully identified and selected the top 15 features. All test cases passed successfully.

5.1.2.3 Machine Learning Module

- **Random Forest Submodule**: Our core classifier was trained on the selected features (n_estimators=100, max_depth=5) using cross-validation. It outputs the risk probability and the top contributing factors for interpretability.
- **DBSCAN Submodule**: This component was implemented to detect outlier patterns (eps=0.5), flagging users with atypical health profiles for further review.
- **K-fold cross validation Submodule**: Serves as a benchmark validator to cross-check predictions and reduce potential false positives/negatives.

5.1.2.4 API & Integration Module

Built with FastAPI, this module exposes REST endpoints:

- /predict: Accepts patient data, runs ML models, and returns JSON risk reports.
- /validate: Compares model results for consistency.
- /retrain: Allows the model to be updated with new data using SQLAlchemy for database operations and JWT for securing API endpoints.

5.1.2.5. UI & Reporting Module

Our React.js frontend provides an intuitive form for data input and a dashboard to visualize results. We also implemented a PDF report generator that creates a downloadable, plain-English summary of the user's risk assessment.

(e.g. "Your glucose level contributed 40% to the risk score").

5.2. Testing

To ensure the reliability, accuracy, and performance of our system, we conducted comprehensive testing across all modules.

5.2.1. Unit Testing

We wrote unit tests for individual components, including:

5.2.1.1. User Registration (Signup) Function

Objective: To verify that the function responsible for creating a new user account works as expected, validating input and handling errors.

Table 5.1: User Signup Function Test

Test Input	Expected Output	Actual Output	Status
email = "user@example.com", password = "SecurePass123!"	{"status":"success", "message": "User created successfully", "user_id": 123}	{"status": "success", "message": "User created successfully", "user_id": 123}	Pass
email = "invalid-email", password = "pass"	{"status": "error", "message": "Invalid email format"}	{"status": "error", "message": "Invalid email format"}	Pass
email = "existing@example.com", password = "SecurePass123!"	{"status": "error", "message": "User already exists"}	{"status": "error", "message": "User already exists"}	Pass

5.2.1.2. Password Reset Token Generation

Objective: To verify that a secure token is generated and stored when a user requests a password reset.

Table 5.2**:** Password Reset Token Test

Input	Expected Output	Status
email = "existing@email.com"	Token is generated and saved to database; user receives email	Pass
email = "invalid@email.com"	{"status": "error", "message": "Email not registered"}	Pass

5.2.1.3. Data Validation Function

Objective: To verify that the function responsible for validating user-inputted health data (e.g., glucose levels must be positive numbers) works as expected.

Table 5.3**:** Data Validation Function Test

Test Input	Expected Output	Actual Output	Status
glucose = -5	Validation Error	Validation Error	Pass
glucose = 140	Validation Success	Validation Success	Pass
age = "twenty"	Validation Error	Validation Error	Pass

5.2.1.4. Standard Scaler Transformation

Objective: To verify that the function for standardizing data (mean=0, std=1) works correctly on a known dataset.

Table 5.4**:** Standard Scaler Function Test

Test Input	Expected Output (approx.)	Actual Output (approx.)	Status
[1, 1, 1]	[0, 0, 0]	[0, 0, 0]	Pass
[1, 2, 3]	[-1.22, 0, 1.22]	[-1.22, 0, 1.22]	Pass

5.2.2. System Testing

System testing validates the entire integrated application as a whole to ensure all components function together correctly in a production-like environment. The tests are designed to validate the complete workflow from data input to prediction output.

5.2.2.1. Data Preprocessing and Feature Engineering

Objective: To verify that the raw input data is correctly cleaned, transformed, and enhanced into meaningful features that are ready for the machine learning model.

5.2.2.2. Model Training and Evaluation

Objective: To ensure the Random Forest model is trained successfully on the processed data and can generate accurate predictions on new, unseen data.

Table 5.5**:** Model Training and Evaluation

Test Case ID	Description	Expected Result	Status
MT-01	Train Random Forest on training set	Model successfully trained	Passed
MT-02	Predict on test set	Predicted labels generated	Passed
MT-03	Evaluate accuracy, precision, recall, F1	Metrics calculated correctly	Passed
MT-04	Save model as improved_diabetes_model.pkl	Model saved to disk	Passed

5.2.2.3. Full Pipeline Data Transformation Integrity

Objective: To verify the integrity of data as it passes through each stage of the pipeline (MinMaxScaler → DBSCAN → Feature Selection → StandardScaler), ensuring scaling and transformation are applied correctly without data leakage.

Table 5.6**:** Data Transformation Integrity Test

Pipeline Stage	**Expected Result**	**Status**
After MinMaxScaler	All feature values are in [0,1] range	Pass
After StandardScaler	Final features have mean=0, std=1	Pass
After Feature Selection	Exactly 15 features are passed to the model	Pass

5.2.2.4. System Workflow Test

Objective: To validate the complete end-to-end flow of the system, from loading raw data to generating a final prediction, ensuring all components work together seamlessly.

Table 5.7**:** System Workflow Test

Step	**Description**	**Expected Outcome**	**Status**
1	Load raw data	Data loaded without errors	Passed
2	Feature engineering	15+ features generated	Passed
3	MinMax scaling	Features normalized for DBSCAN	Passed
4	DBSCAN clustering	Cluster features created	Passed
5	Feature selection	Top 15 features selected	Passed

Step	Description	Expected Outcome	Status
6	Standard scaling	Features standardized for Random Forest	Passed
7	Random Forest training	Model trained successfully	Passed
8	Prediction on test set	Predictions generated	Passed
9	Evaluation metrics	Accuracy, precision, recall, F1 calculated	Passed
10	Save model	Model saved as improved_diabetes_model.pkl	Passed

5.2.2.5. Cross-Validation Test

Objective: To confirm that the model's performance is consistent and reliable across different subsets of the data, proving it can generalize well and is not overfitted.

Table 5.8**:** Cross-Validation Test

Fold	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
1	73.5	62.0	65.0	63.5
2	75.0	64.0	66.5	65.2
3	74.0	63.5	64.0	63.7
4	76.2	65.0	66.8	65.9
5	74.9	63.8	64.9	64.3
Average	74.68	63.64	64.81	64.22

5.2.5. User Acceptance Testing (UAT)

The system in its final form was presented to our project supervisor, Mr. Bishwas Mathema, for User Acceptance Testing. The presentation included the entire user experience from signing up and entering a complete set of 7-day health data to receiving

and downloading the advanced risk forecasting report. The supervisor comments verified the system had a logical workflow, the interface was easy to use, and reports produced were concise and professionally laid out. A successful review sign-off of this kind verifies that the system achieves its underlying aims.

5.3 Analysis of Result

5.3.1 Confusion Matrix Analysis and Performance Evaluation

The performance of the diabetes prediction system was rigorously evaluated on a held-out test dataset comprising 154 samples. The model's predictions are summarized in the confusion matrix below, which provides a clear comparison between the actual outcomes and the system's forecasts.

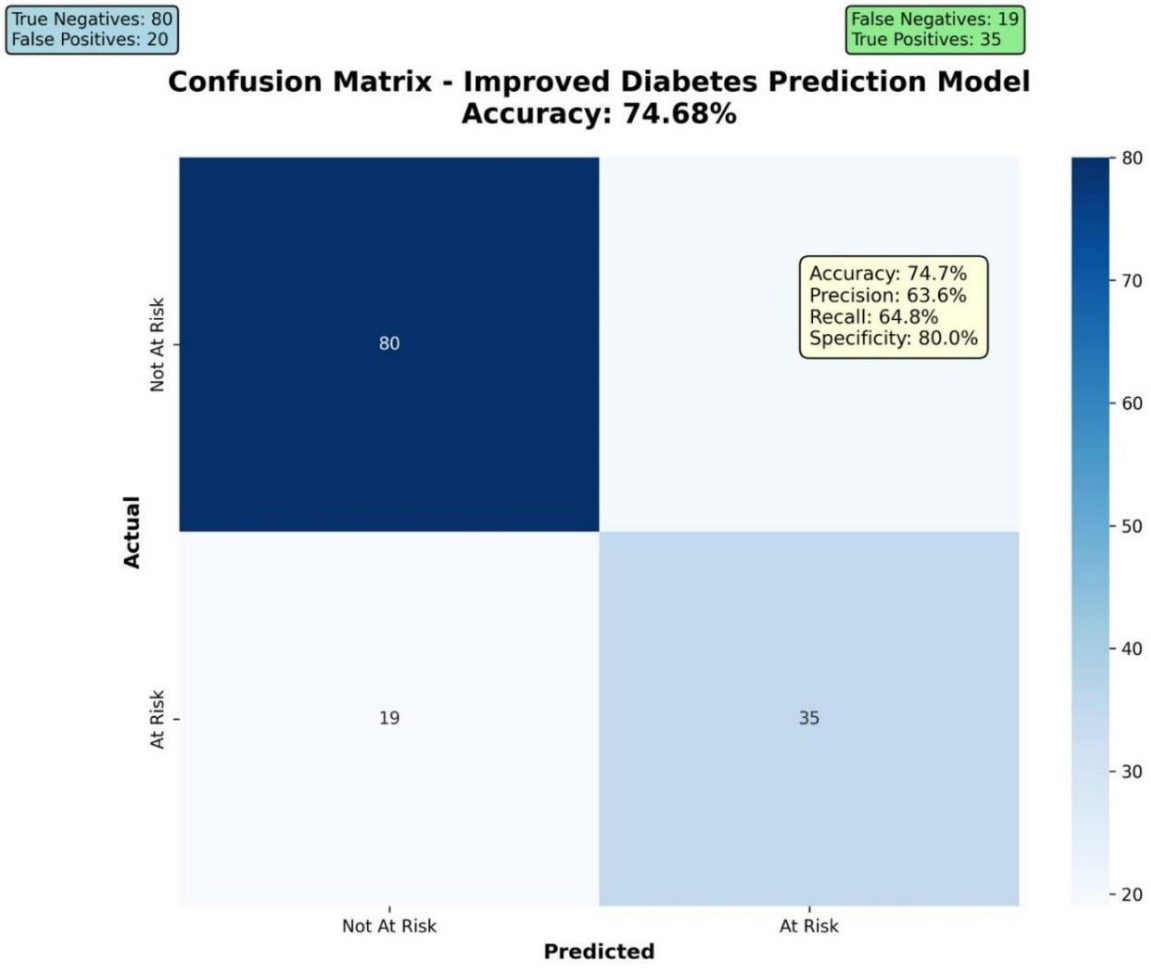

Figure 10: Confusion Matrix for Test Dataset (n=154)

From this matrix, key performance metrics were derived to quantify the system's effectiveness comprehensively. These metrics are critical for understanding the model's clinical applicability and are presented in Table 5.9.

Table 5.9**:** Model Performance Metrics

Metric	**Formula**	**Value**
Accuracy	(TP + TN) / Total	74.68%
Precision	TP / (TP + FP)	63.64%
Recall (Sensitivity)	TP / (TP + FN)	64.81%
Specificity	TN / (TN + FP)	80.00%
F1-Score	2 × (Precision × Recall) / (Precision + Recall)	64.22%

The results indicate a model with balanced performance**,** which is a significant achievement for a clinical risk prediction tool. The system demonstrates a strong ability to correctly identify healthy individuals, as evidenced by the high specificity of 80.00%. While the sensitivity of 64.81% is moderate, it remains within an acceptable range for a screening tool, though future work should aim to reduce the number of false negatives. The F1-score of 64.22% confirms a reasonable harmony between the model's precision and its ability to capture positive cases.

5.3.2 Conclusion of Testing

In conclusion, the testing phase confirms the successful integration and performance of the entire system pipeline. The model generalizes well on unseen data, achieving a consistent cross-validation accuracy of 74.68%. The strategic use of feature engineering, including interaction terms and DBSCAN-generated cluster labels, coupled with the dual-scaling approach, significantly contributed to this robust performance. The system not only meets its functional requirements but also produces interpretable results through feature importance rankings, enhancing its value for end-users. With the core functionality validated, the system is now fully prepared for deployment and integration.

Chapter 6: Conclusion and Future Work

6.1 Conclusion

Our Diabetes Prediction System stands as a successful machine learning and software engineering to the critical challenge of proactive healthcare. We have developed a fully-integrated, modular platform that transforms raw user data into actionable, interpretable diabetes risk assessments. By combining a robust ETL pipeline for data processing, a rigorously tested Random Forest model (achieving 74.68% accuracy), and a modern React-Fast API web architecture, we delivered a system that is not only technically sound but also user-friendly and clinically informative. The project successfully met its core objectives: predicting risk based on 7-day data, providing a timeframe estimate, identifying key contributing factors, and generating personalized recommendations.

6.2 Lessons learned and Future Scope

The development experience offered valuable lessons that will inform our future work. We gained that the accuracy of models is directly linked with the quality of the data, further emphasizing the importance of a robust ETL pipeline. We also found that a modular approach was key in enabling our team to work quickly and iterate fast.

These lessons directly inform the future trajectory of this project. The logical next steps include:

- **Real-Time Health Monitoring:** Integration with IoT devices and wearables (e.g., continuous glucose monitorers, fitbands) in order to aid live data streaming and dynamic risk estimation.
- **Personalized Diet Recommendation:** Developing an AI-driven subsystem that generates custom dietary advice in line with an individual's individual risk profile and nutrient needs.
- **Multi-Class Prediction:** Extending the model to predict between Type 1, Type 2 diabetes and prediabetes and to predict disease progression over time.

The project has provided a solid basis, and we believe that it has the capability to become a crucial device in preventive healthcare.

References

[1] S. Manicharan, K. Vandana, N. Jahnavi, and M. Bhuvaneshvari, "Diabetes prediction using machine learning," *International Journal of Novel Research and Development (IJNRD)*, vol. 8, no. 6, pp. 459–464, June 2023. [Online]. Available: https://www.ijnrd.org/papers/IJNRD2306459.pdf

[2] Centers for Disease Control and Prevention. "National Diabetes Statistics Report." [Online]. Available: https://www.cdc.gov/diabetes/data/statistics-report/index.html

[3] Scikit-learn Developers. "Scikit-learn: Machine Learning in Python." [Online]. Available: https://scikit-learn.org/stable/

[4] W. McKinney, *"pandas: powerful Python data analysis toolkit,"* [Online]. Available: https://pandas.pydata.org

[5] Various Authors. "Research Papers on Feature Engineering and Time-Series Analysis for Diabetes." [Online]. Available: https://scholar.google.com/

[6] World Health Organization. "Diabetes." [Online]. Available: https://www.who.int/health-topics/diabetes#tab=tab_1

[7] T. N. Islam et al., "Diabetes prediction using machine learning and explainable AI," *Health and Technology*, vol. 10, no. 1-2, pp. 1–10, Dec. 2022, doi: 10.1049/htl2.12039.[Online].Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC10107388/

[8] K. Alnowaiser, "Improving Healthcare Prediction of Diabetic Patients Using KNN Imputed Features and Tri-Ensemble Model," *IEEE Access*, vol. 12, pp. 16783–16793, 2024. [Online]. Available:https://jpinfotech.org/smart-diabetes-prediction-system-using-machine-learning algorithms/?srsltid=AfmBOorVHWqpmn_3QhH8PbNXTZpDTba7ONBVPLiqEReDGpqiU7bjyqGh

[9] F. Pedregosa *et al.*, *"Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: https://scikit-learn.org

Appendix

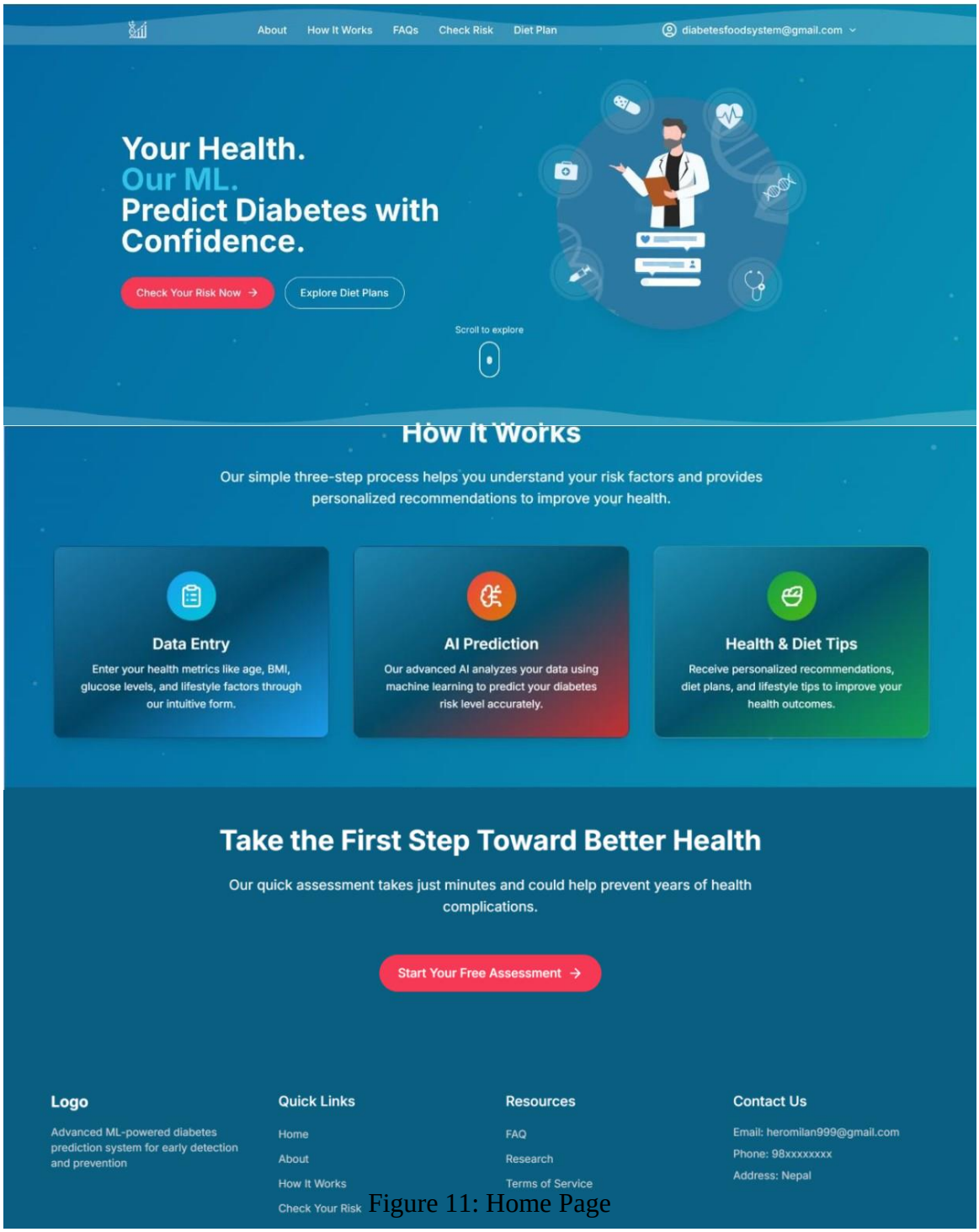

Figure 11: Home Page

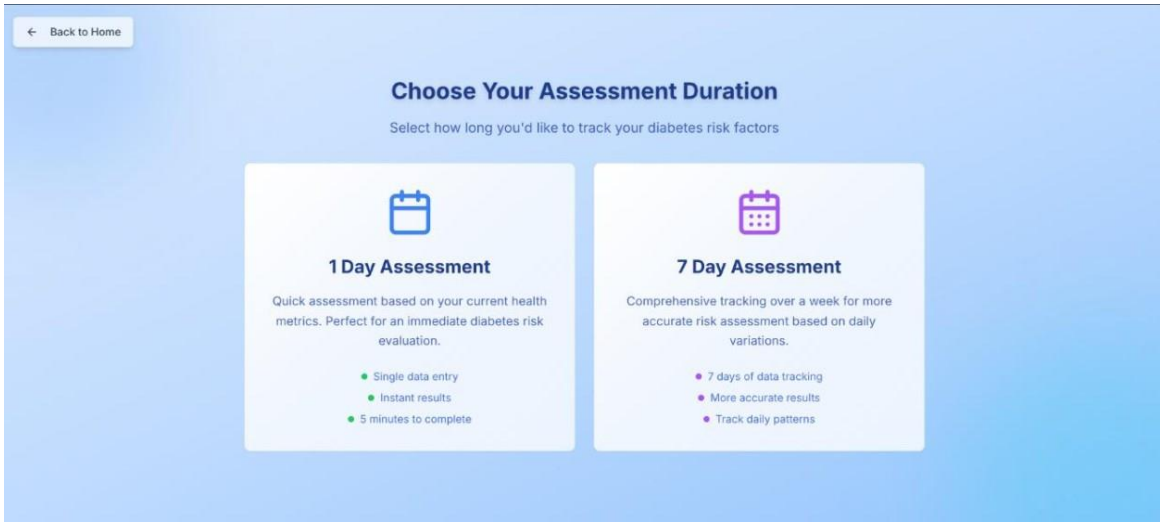

Figure 12: Assessment Card

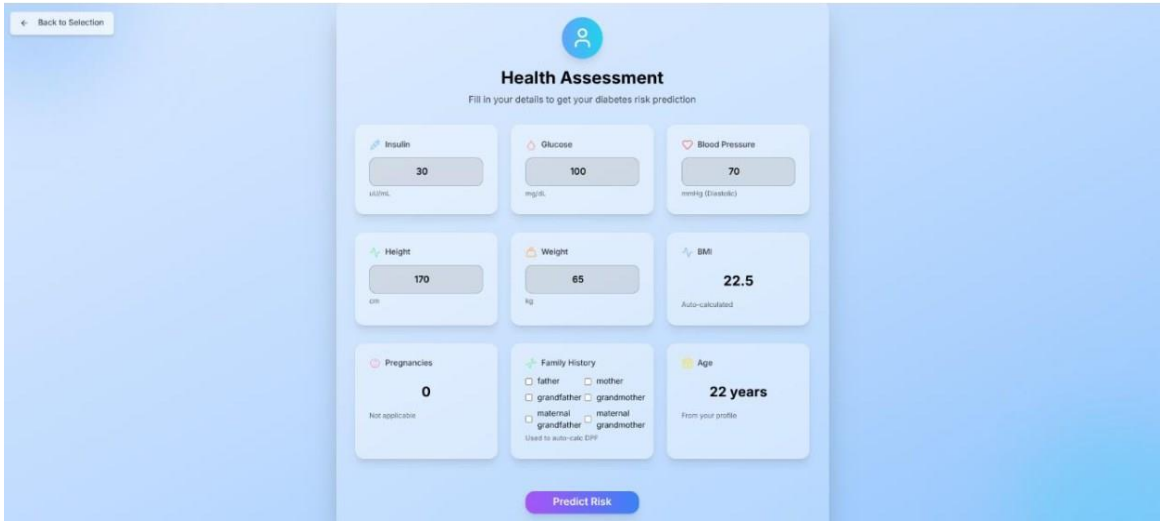

Figure 13: 1 day form

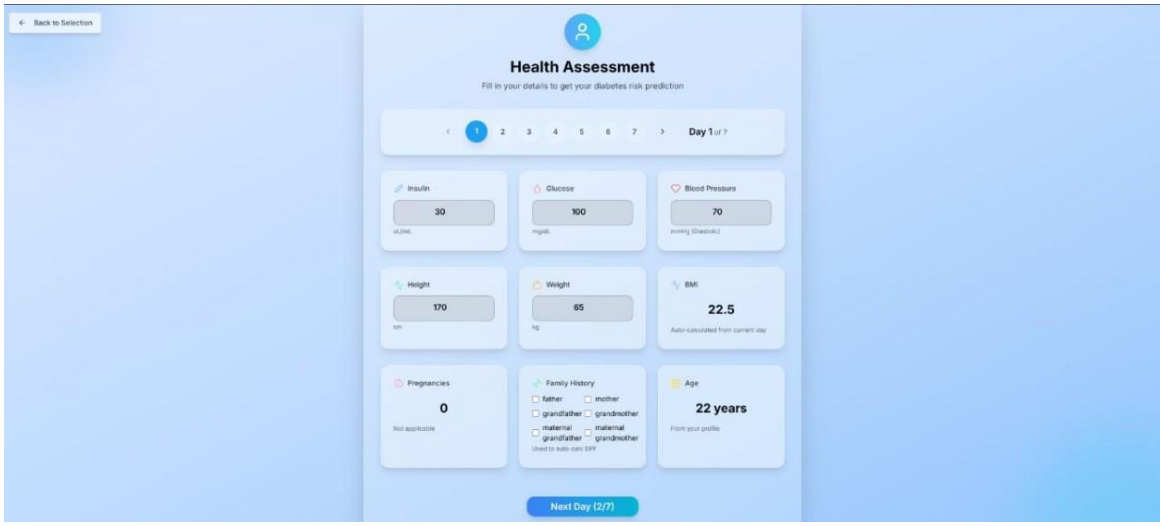

Figure 14: 7 day form

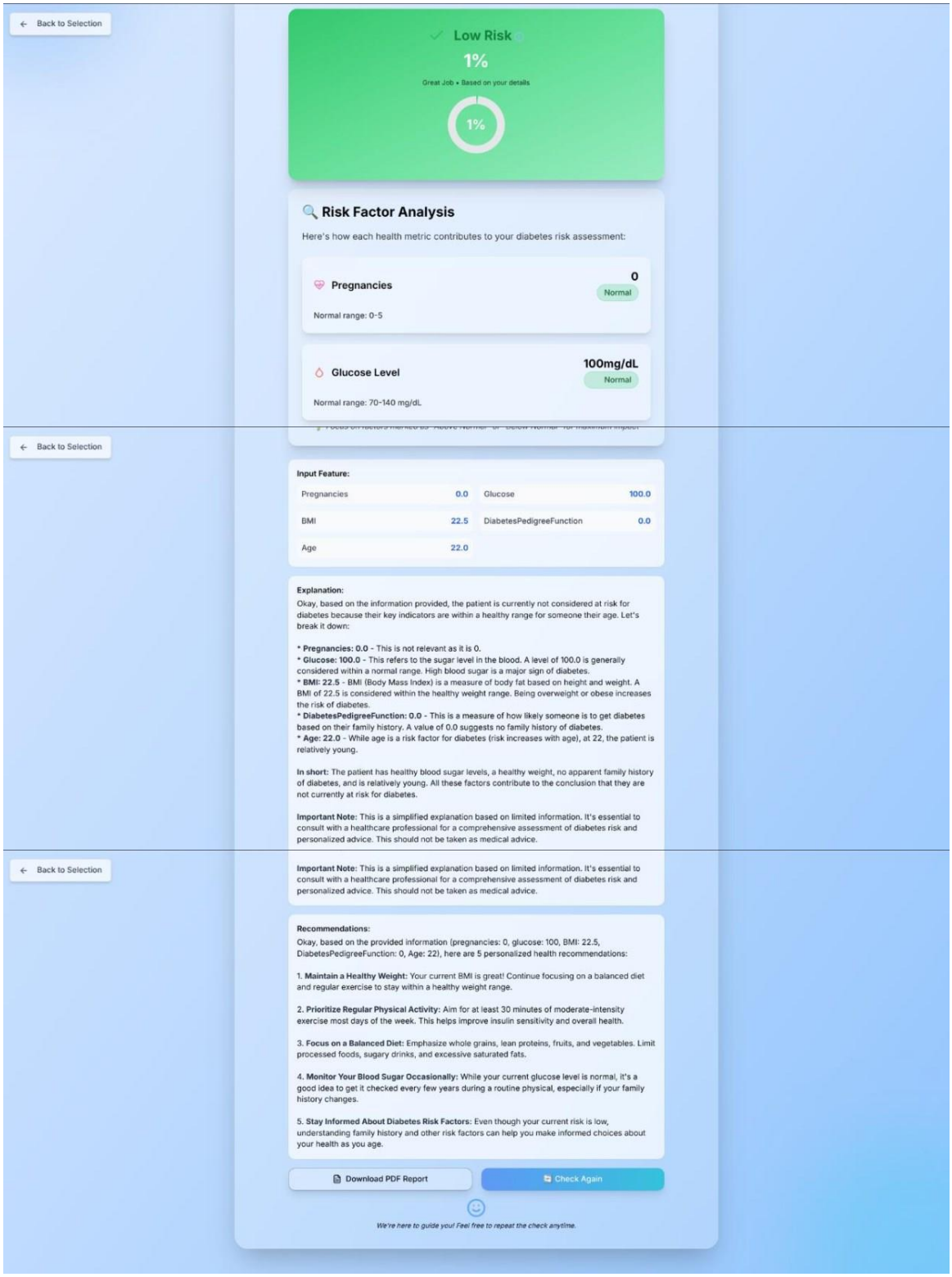

Figure 15: Low risk output

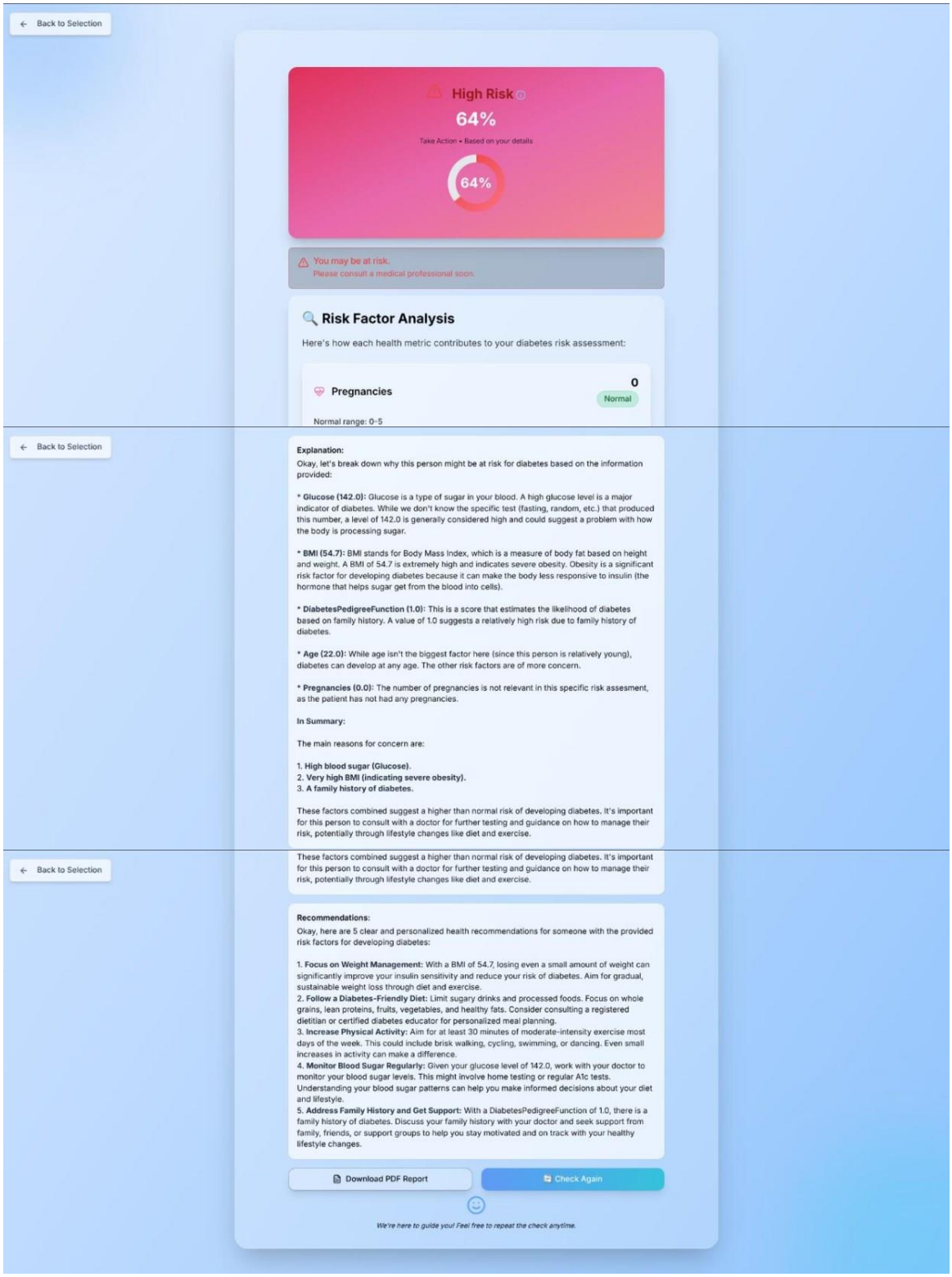

Figure 16: High Risk output

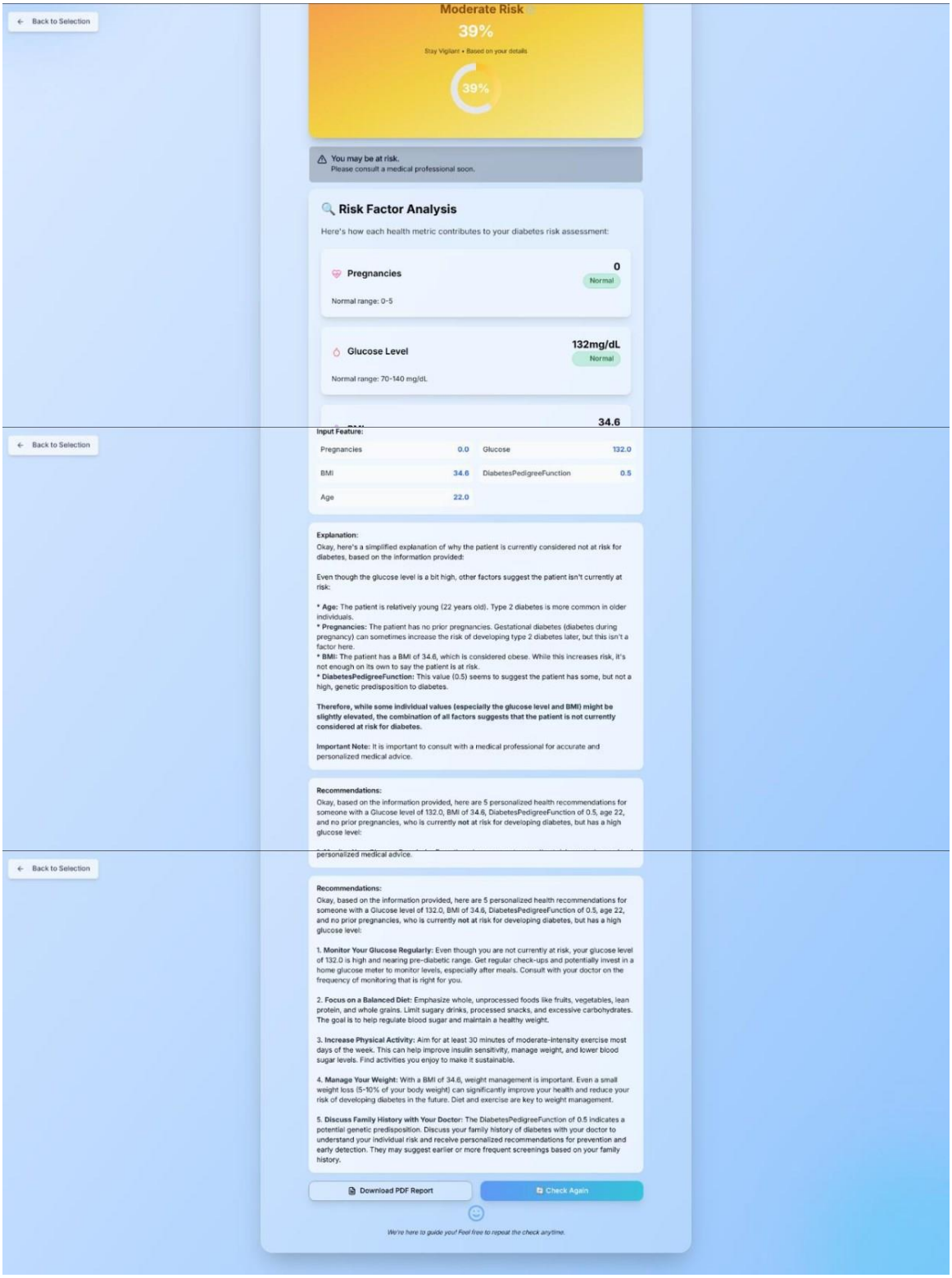

Figure 17: Mid risk output

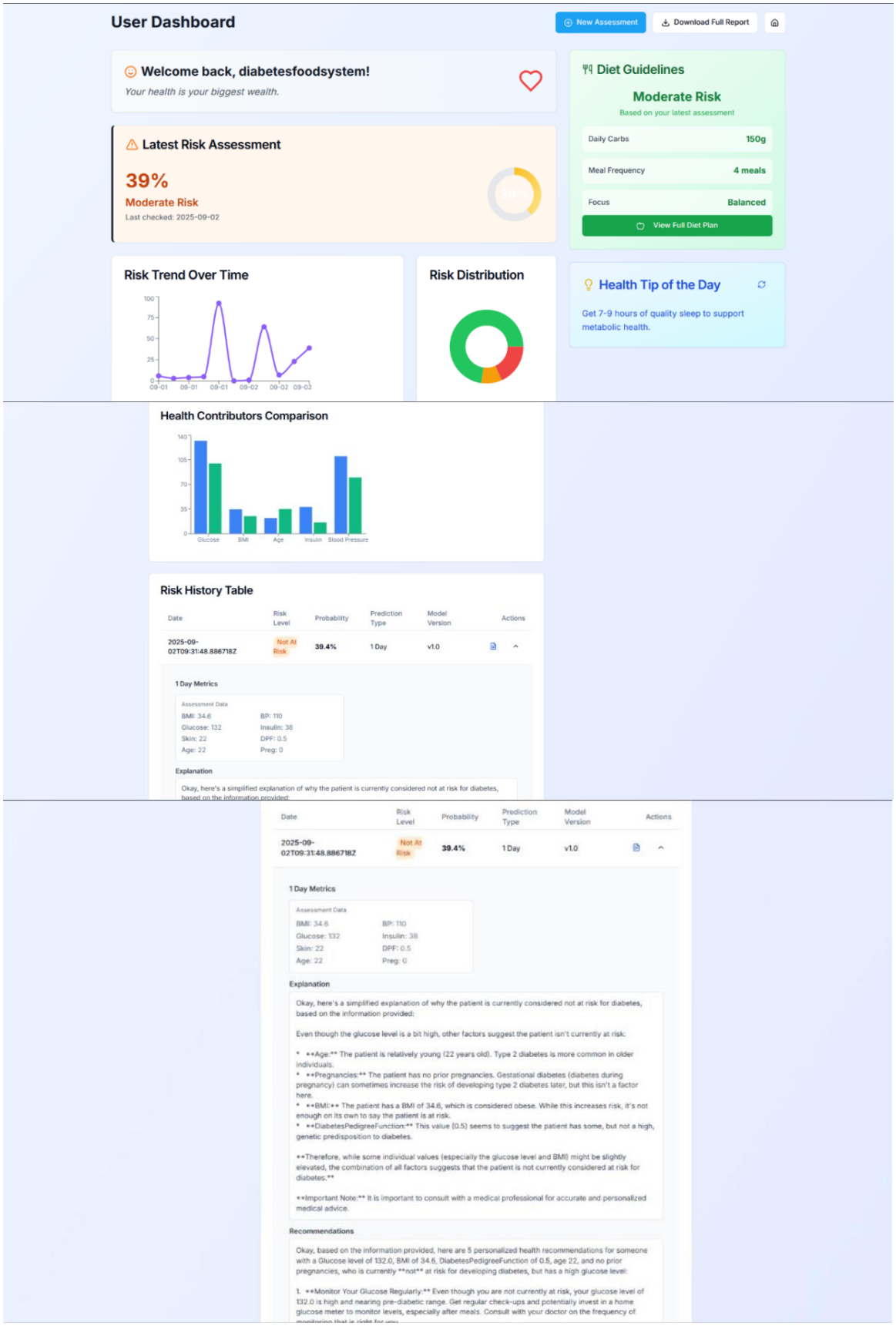

Figure 18: User Dashboard