Project Report

# HEALTH CHECK

**REALIZED BY:**

Anass BOUABBOUS

Hassan FOURKA

Mohamed Reda TAOUS

**SUPERVISED BY:**

HAFIDI Hakim

GAMOUH Hamza

ADERGHAL Yasser

Table des matières

[HEALTH CHECK 1](#_Toc187818234)

[Front Matter 6](#_Toc187818235)

[Acknowledgments 6](#_Toc187818236)

[Abstract 6](#_Toc187818237)

[Introduction 6](#_Toc187818238)

[Background 7](#_Toc187818239)

[Introduction to the Problem : 7](#_Toc187818240)

[Importance of the Problem 7](#_Toc187818241)

[Existing Solutions 7](#_Toc187818242)

[Related Work 7](#_Toc187818243)

[Novelty of the Approach 8](#_Toc187818244)

[Who Benefits? 8](#_Toc187818245)

[Relevance to Other Work 8](#_Toc187818246)

[Requirements Capture 8](#_Toc187818247)

[Functional Requirements 8](#_Toc187818248)

[Analysis and Design 9](#_Toc187818249)

[Project Overview 9](#_Toc187818250)

[High-Level System Design 9](#_Toc187818251)

[The system was designed with the following components: 9](#_Toc187818252)

[Frontend: 9](#_Toc187818253)

[Backend: 9](#_Toc187818254)

[Machine Learning Models: 9](#_Toc187818255)

[Integration and Deployment: 10](#_Toc187818256)

[Key Design Choices 10](#_Toc187818257)

[User-Centric Design 10](#_Toc187818258)

[Choice of Machine Learning Models 10](#_Toc187818259)

[Lightweight Framework for Backend 10](#_Toc187818260)

[Modular Design 10](#_Toc187818261)

[Challenges and Iterations 10](#_Toc187818262)

[Initial Design and Challenges 10](#_Toc187818263)

[User Experience Enhancements 11](#_Toc187818264)

[Feedback from early testers indicated a need for: 11](#_Toc187818265)

[Machine Learning Iterations 11](#_Toc187818266)

[Parts Implemented and Limitations 11](#_Toc187818267)

[Implemented: 11](#_Toc187818268)

[Not Implemented: 11](#_Toc187818269)

[Trade-Offs and Design Decisions 11](#_Toc187818270)

[Accuracy vs. Complexity: 11](#_Toc187818271)

[Performance vs. Features: 11](#_Toc187818272)

[Local vs. Cloud Deployment: 11](#_Toc187818273)

[Implementation 12](#_Toc187818274)

[Overview 12](#_Toc187818275)

[Backend Implementation 12](#_Toc187818276)

[Key Code Snippet: Flask Routes 12](#_Toc187818277)

[Key Features: 12](#_Toc187818278)

[Machine Learning Integration 13](#_Toc187818279)

[Frontend Implementation 13](#_Toc187818280)

[Challenges and Interesting Aspects 13](#_Toc187818281)

[1. Model Deployment 13](#_Toc187818282)

[2. User Experience 13](#_Toc187818283)

[3. Backend-Frontend Communication 13](#_Toc187818284)

[Testing 13](#_Toc187818285)

[Overview 13](#_Toc187818286)

[Test Plan 14](#_Toc187818287)

[Objectives 14](#_Toc187818288)

[Methods 14](#_Toc187818289)

[Backend Testing: 14](#_Toc187818290)

[Frontend Testing: 14](#_Toc187818291)

[End-to-End Testing: 14](#_Toc187818292)

[Results 15](#_Toc187818293)

[1. Machine Learning Model Testing 15](#_Toc187818294)

[2. Backend Testing 15](#_Toc187818295)

[3. Frontend Testing 15](#_Toc187818296)

[4. End-to-End Testing 16](#_Toc187818297)

[Evaluation 16](#_Toc187818298)

[Fulfillment of Objectives 16](#_Toc187818299)

[Advantages of This Approach: 16](#_Toc187818300)

[Integration of Machine Learning: 16](#_Toc187818301)

[Accessibility: 16](#_Toc187818302)

[User-Centric Design: 17](#_Toc187818303)

[Scalability: 17](#_Toc187818304)

[Disadvantages Compared to Related Work: 17](#_Toc187818305)

[Limited Disease Coverage: 17](#_Toc187818306)

[Cloud Deployment: 17](#_Toc187818307)

[Dataset Diversity: 17](#_Toc187818308)

[Strengths and Weaknesses of the Approach 17](#_Toc187818309)

[Strengths: 17](#_Toc187818310)

[Weaknesses: 17](#_Toc187818311)

[Reflection on Design Decisions 18](#_Toc187818312)

[Significance of the Work 18](#_Toc187818313)

[Summary of Lessons Learned 18](#_Toc187818314)

[Conclusions and Further Work 19](#_Toc187818315)

[Summary of Achievements 19](#_Toc187818316)

[Design Choices and Rationale 19](#_Toc187818317)

[Challenges and Difficulties 19](#_Toc187818318)

[Novel Contributions 20](#_Toc187818319)

[Lessons Learned 20](#_Toc187818320)

[Further Work 20](#_Toc187818321)

[User Guide : 21](#_Toc187818322)

[About page: 22](#_Toc187818323)

[To do a Diabetes Prediction 22](#_Toc187818324)

[Bibliography 29](#_Toc187818325)

[Scikit-learn Documentation 29](#_Toc187818326)

[Flask Framework Documentation 29](#_Toc187818327)

[TensorFlow Documentation 29](#_Toc187818328)

[Heart Disease Dataset 29](#_Toc187818329)

[Bootstrap Documentation 29](#_Toc187818330)

[Joblib Documentation 29](#_Toc187818331)

[Kaggle for Data Hosting and Analysis 29](#_Toc187818332)

[Appendices 30](#_Toc187818333)

[Test Data Sets 30](#_Toc187818334)

[Raw Results 30](#_Toc187818335)

[Related Resource 31](#_Toc187818336)

## Front Matter

Title: Health Prediction System Using AI and Machine Learning  
Submitted by: [Taous Mohamed Reda,Bouabbous Anass,Fourka Hassan]  
Supervisor: [ADERGHAL YASSER,HAFIDI HAKIM]  
Date: [15 /01 /2025]

## Acknowledgments

I would like to express my sincere gratitude to my supervisor, [ADERGHAL YASSER,HAFIDI HAKIM], for their invaluable guidance and support throughout the duration of this project. I also thank my colleagues, friends, and family for their encouragement and assistance. This project would not have been possible without their help.

## Abstract

This report presents the development of a Health Prediction System designed to assist in early detection of diabetes and heart disease using artificial intelligence and machine learning. The system integrates a Flask-based backend, a Bootstrap-enhanced frontend, and predictive models trained on medical datasets. The project aims to provide users with a seamless and accurate health prediction experience through a user-friendly web interface. Testing and evaluation demonstrate the potential of the system in supporting early health diagnosis, highlighting areas for further improvement and development.

## Introduction

The Health Prediction System is designed to address the growing need for predictive healthcare tools. With rising incidences of chronic illnesses like diabetes and heart disease, early detection is crucial. This system leverages the power of AI and machine learning to predict the likelihood of these diseases based on user-provided data. The report outlines the system's development process, from problem identification to implementation and testing, emphasizing its significance in modern healthcare.

## Background

### Introduction to the Problem :

Diabetes and heart disease are among the leading causes of death worldwide, with millions of people suffering from these conditions. Early detection and management of these diseases are critical to improving patient outcomes and reducing the burden on healthcare systems. However, traditional diagnostic methods can often be time-consuming, expensive, and inaccessible to many individuals. This project seeks to address these challenges by developing an interactive web application that provides predictive health analysis for diabetes and heart disease.

### Importance of the Problem

The importance of this problem lies in its global impact. According to the World Health Organization (WHO), the prevalence of diabetes and heart disease continues to rise due to factors such as sedentary lifestyles, poor diet, and aging populations. Early detection tools like the one proposed in this project can empower individuals to take preventive actions, seek timely medical attention, and manage their health more effectively. This is especially valuable in remote or underserved areas where access to healthcare professionals may be limited.

### Existing Solutions

Numerous applications and tools exist for health monitoring and disease prediction. For example:

1. **Mobile Health Apps:** Applications like MyFitnessPal and HealthifyMe offer lifestyle management features but lack predictive capabilities specific to diabetes or heart disease.
2. **Machine Learning Models:** Research in artificial intelligence has produced predictive models for diagnosing diseases, but these are often limited to academic or clinical environments and are not readily accessible to the general public.
3. **Wearable Devices:** Devices like Fitbit and Apple Watch provide health data but do not offer specialized disease prediction tools.

While these solutions contribute to health monitoring, they often lack an integrated, user-friendly interface that combines predictive analytics with actionable insights, which is the focus of this project.

### Related Work

This project builds upon advancements in:

1. **Machine Learning:** Techniques such as Random Forest and Neural Networks have been used to predict diseases with high accuracy. Studies have demonstrated their efficacy in analyzing patient data to detect conditions like diabetes and heart disease.
2. **Web Development:** Modern frameworks like Flask enable the integration of machine learning models into web applications, making predictive tools accessible to end-users.
3. **Healthcare Informatics:** The use of datasets like the Pima Indians Diabetes Database and heart disease datasets has been critical in training models for accurate prediction.

### Novelty of the Approach

The Health Check web application distinguishes itself by integrating machine learning models into an intuitive, web-based platform. Unlike other tools, it:

* Focuses specifically on diabetes and heart disease predictions.
* Provides a seamless user experience with an easy-to-navigate interface.
* Combines backend machine learning models with frontend interactivity for real-time predictions.
* Allows users to input basic health metrics and instantly receive insights about their health status.

### Who Benefits?

The primary beneficiaries of this project are:

* **Individuals at Risk:** People who may suspect they have symptoms but lack access to healthcare facilities.
* **Healthcare Professionals:** Doctors can use this tool as a preliminary assessment to guide further diagnostics.
* **Public Health Initiatives:** Organizations can leverage such tools for community screenings and early interventions.

### Relevance to Other Work

This project contributes to ongoing efforts in preventive healthcare and machine learning. It builds on prior research in disease prediction algorithms while addressing the gap in accessibility and usability. By bridging the divide between academic research and real-world application, the Health Check web application aims to make a meaningful impact in healthcare technology.

## Requirements Capture

### Functional Requirements

* **The system must allow users to input health-related data (e.g., glucose level, blood pressure).**
* **The system must process this data and provide accurate predictions for diabetes and heart disease.**
* **The system must display results clearly and concisely.**

**Non-Functional Requirements**

* **The system should be responsive and work on multiple devices.**
* **Predictions should be generated within a few seconds.**
* **The backend should be scalable to accommodate future enhancements.**

## Analysis and Design

### Project Overview

**The goal of the Health Check application is to provide users with an intuitive and accessible web-based platform to predict the likelihood of diabetes and heart disease using machine learning models. The design process involved integrating machine learning, backend development, and frontend design to create a cohesive system that ensures functionality, usability, and scalability.**

## High-Level System Design

### The system was designed with the following components:

## Frontend:

* + **Built using HTML, CSS, and JavaScript with Bootstrap for responsive design.**
  + **A clean and intuitive user interface allows users to input their health metrics and receive instant results.**
  + **Dark mode was implemented to improve user experience for those who prefer a low-light interface.**

## Backend:

* + **Developed using Flask, a lightweight Python web framework.**
  + **Manages routing, user input validation, and communication between the frontend and the machine learning models.**
  + **Ensures secure handling of user data.**

### Machine Learning Models:

* + **Two separate models were implemented:**
    - **Diabetes Prediction Model: A Random Forest Classifier trained on the Pima Indians Diabetes Database.**
    - **Heart Disease Prediction Model: A Neural Network model trained on a heart disease dataset.**
  + **Models were trained, optimized, and serialized using libraries like scikit-learn and TensorFlow for efficient deployment.**
  + **Input data is processed and normalized to match the training datasets before being passed to the models.**

### Integration and Deployment:

* + **The machine learning models are integrated into the Flask backend, exposing them via API endpoints.**
  + **Deployed locally for testing and demonstration purposes, with the potential to be hosted on platforms like AWS or Heroku.**

## Key Design Choices

### User-Centric Design

**A priority was given to creating a user-friendly interface. The layout is responsive and straightforward, with clear instructions on data input and immediate feedback for predictions. User interaction elements such as buttons and forms were styled to provide a professional appearance.**

### Choice of Machine Learning Models

* **The Random Forest Classifier was chosen for diabetes prediction because of its ability to handle imbalanced datasets and provide robust results.**
* **The Neural Network was selected for heart disease prediction due to its capability to handle complex patterns in the data.**

### Lightweight Framework for Backend

**Flask was chosen for its simplicity and ability to integrate seamlessly with Python-based machine learning libraries. It provided a scalable and efficient way to serve the models via RESTful APIs.**

### Modular Design

**The system was built using a modular approach, allowing individual components (frontend, backend, models) to be updated or replaced without significant impact on the other components.**

## Challenges and Iterations

### Initial Design and Challenges

**The initial design aimed to deploy models via a cloud platform; however, local deployment was chosen due to time constraints and to ensure a functional demonstration. The need for secure handling of user data also influenced the decision to keep deployment local during testing phases.**

## User Experience Enhancements

### Feedback from early testers indicated a need for:

* A more engaging user interface, leading to the addition of dark mode and interactive buttons.
* Clearer instructions for input fields, which were addressed by adding placeholder text and tooltips.

### Machine Learning Iterations

Initial model predictions lacked the desired accuracy. Model optimization techniques such as hyperparameter tuning and cross-validation were applied to improve results. The feature importance of the diabetes model was analyzed, leading to refined input fields that better matched the model's strengths.

## Parts Implemented and Limitations

### Implemented:

* Full integration of both models into the web application.
* A responsive and interactive frontend with dark mode support.
* Backend routing for handling user input and serving model predictions.

### Not Implemented:

* Live deployment on a cloud platform due to time constraints.
* Advanced user features such as saving prediction history or generating detailed reports.

## Trade-Offs and Design Decisions

### Accuracy vs. Complexity:

* + Simpler models like Random Forest were chosen for diabetes prediction to ensure interpretability, sacrificing some potential gains in accuracy compared to more complex models.

### Performance vs. Features:

* + A minimal feature set was implemented to prioritize stability and responsiveness over adding non-essential functionalities like user accounts or dashboards.

### Local vs. Cloud Deployment:

* + The decision to deploy locally was driven by time constraints and to maintain control over the application during development.

## Implementation

### Overview

The implementation of the Health Check project involved developing a full-stack web application that integrates machine learning models for health predictions. The backend system was developed using Flask, while the frontend interface was designed using HTML, CSS, and JavaScript with Bootstrap for responsiveness. The machine learning models were trained using scikit-learn and TensorFlow and deployed via the Flask backend.

This section highlights the key aspects of the implementation, including interesting technical challenges and decisions. Code snippets are provided to illustrate the algorithmic flow and critical interactions between components.

### Backend Implementation

The backend is built using Flask and serves as the backbone of the application. It manages routing, input validation, and the integration of machine learning models.

### Key Code Snippet: Flask Routes

Below is a simplified snippet demonstrating how the Flask app handles routing for the prediction endpoints:

**Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement**

### Key Features:

* Data validation and parsing from user input.
* Integration of the trained Random Forest Classifier (diabetes\_model) for predictions.
* Conditional rendering of results based on the prediction outcome.

### Machine Learning Integration

Two machine-learning models were integrated into the application:

1. **Diabetes Prediction Model**: A Random Forest Classifier trained on the Pima Indians Diabetes Database.
2. **Heart Disease Prediction Model**: A Neural Network trained on the UCI Heart Disease dataset.

### Frontend Implementation

The frontend interface was developed using HTML templates with Jinja2 for dynamic rendering, Bootstrap for styling, and JavaScript for interactivity. It includes:

* **Input Forms**: These are used to collect user health metrics.
* **Prediction Results Page**: Displays personalized feedback based on the model’s output.

## Challenges and Interesting Aspects

### 1. Model Deployment

The integration of machine learning models into a web environment requires:

* Normalizing input data to match the models' training datasets.
* Handling exceptions gracefully when user inputs were invalid or incomplete.

### 2. User Experience

To enhance user interaction, features like dark mode, hover effects, and responsive layouts were implemented. These required additional CSS and JavaScript modifications.

### 3. Backend-Frontend Communication

Ensuring seamless data flow between the frontend and backend, especially when dealing with diverse data types (e.g., integers, floats), posed an initial challenge but was resolved through rigorous validation and error handling.

## Testing

### Overview

The primary goal of the testing phase was to ensure the Health Check application functionsas intended, providing accurate predictions, a seamless user experience, and robust performance under various scenarios. Testing involved the backend (machine learning models and Flask routing) and the frontend (user interface and responsiveness).

The testing process followed a structured approach, which included:

1. Unit Testing for individual components (e.g., prediction algorithms, input validation).
2. Integration Testing for the interactions between backend, frontend, and machine learning models.

## Test Plan

### Objectives

* Verify the correctness of machine learning predictions for diabetes and heart disease.
* Ensure the frontend interface is responsive and user-friendly.
* Test error handling and validation for user inputs.
* Confirm performance stability under different load conditions.

## Methods

### Backend Testing:

* + Test predictions for known cases (e.g., edge cases and common scenarios).
  + Check Flask route responses for valid and invalid inputs.
  + Validate exception handling for invalid data or missing values.

### Frontend Testing:

* + Verify the display and alignment of input forms and results pages.
  + Test responsiveness on various devices (mobile, tablet, desktop).
  + Ensure dark mode and interactive elements function as expected.

### End-to-End Testing:

* + Simulate real user interactions by inputting data and verifying predictions.
  + Check transitions between pages and response times.

## Results

### 1. Machine Learning Model Testing

The models were tested using a set of known test data points from their respective datasets. Accuracy and F1-score metrics were used to evaluate their performance.

| **Model** | **Accuracy** | **F1 Score** |
| --- | --- | --- |
| Diabetes Model | 78% | 0.75 |
| Heart Disease Model | 85% | 0.83 |

#### **Key Observations:**

* Predictions aligned with expected outcomes for both models.
* Slight misclassifications were observed in borderline cases, especially for the diabetes model

### 2. Backend Testing

| **Test Case** | **Expected Outcome** | **Result** |
| --- | --- | --- |
| Valid user input for diabetes | Accurate prediction (Positive/Negative) | Passed |
| Missing input values | Redirect with an error message | Passed |
| Invalid input types (e.g., text for numeric fields) | Redirect with an error message | Passed |

### 3. Frontend Testing

| **Test Case** | **Expected Outcome** | **Result** |
| --- | --- | --- |
| Form alignment on different devices | Proper alignment and responsiveness | Passed |
| Dark mode functionality | UI switches seamlessly to the dark mode | Passed |
| Navigation between pages | Smooth transitions with correct routing | Passed |

### 4. End-to-End Testing

End-to-end testing involved simulating typical user workflows:

1. A user navigates to the Diabetes Prediction page.
2. Inputs test data and submits it.
3. The application returns a prediction and displays results on a separate page.

## Evaluation

### Fulfillment of Objectives

The primary objectives of this project were:

1. Develop a web application integrating AI models for predicting diabetes and heart disease.
2. Ensure accessibility with a user-friendly interface and responsive design.
3. Achieve high accuracy in disease prediction using machine learning models.

These objectives were largely fulfilled:

* A functional web application was successfully developed, incorporating Flask for backend operations and Bootstrap for frontend design.
* The application is accessible on multiple devices, offering a seamless and intuitive user experience.
* The machine learning models provided reliable predictions, with the heart disease model achieving 85% accuracy and the diabetes model achieving 78% accuracy.

Changes in Scope

During the project, minor adjustments were made to the initial scope:

* Deployment: While the initial plan included deploying the application on a cloud platform, this was deferred due to time constraints. Instead, the application was tested locally, ensuring functionality before wider deployment.
* Features: Advanced features, such as user account management or prediction history tracking, were postponed to focus on core functionalities.

These changes allowed the team to prioritize the core objectives of prediction accuracy and usability while leaving room for future development.

## Advantages of This Approach:

### Integration of Machine Learning:

* + Many existing tools focus on lifestyle tracking without offering disease-specific predictions. This project bridges that gap by providing disease-specific predictive analysis.

### Accessibility:

* + Unlike academic tools or complex algorithms limited to research environments, this application is designed for end-users with no technical background.

### User-Centric Design:

* + Features like dark mode, interactive forms, and responsive layouts enhance the user experience, making it more engaging than traditional medical tools.

### Scalability:

* + The modular architecture allows future enhancements, such as adding predictions for more diseases or deploying models on cloud platforms.

## Disadvantages Compared to Related Work:

### Limited Disease Coverage:

* + The application currently supports predictions for only two conditions (diabetes and heart disease), whereas some advanced platforms address multiple diseases.

### Cloud Deployment:

* + Many modern tools leverage cloud platforms for real-time predictions and data analytics. This project currently operates locally, limiting its scalability.

### Dataset Diversity:

* + The models were trained on datasets like the Pima Indians Diabetes Database, which might not generalize well across diverse populations. Further testing on more representative datasets is required.

## Strengths and Weaknesses of the Approach

### Strengths:

#### **Modularity:**

* + The architecture separates the frontend, backend, and machine learning components, simplifying updates and scalability.

#### **Accuracy and Speed:**

* + The models produce results within seconds, ensuring a smooth user experience.

#### **Adaptability:**

* + The system can easily incorporate new models or features without significant rework.

### Weaknesses:

#### **Reliance on Static Data:**

* + Predictions are limited to static user inputs rather than real-time health monitoring.

#### **Lack of Advanced Features:**

* + Features like account management, historical data analysis, or integration with wearable devices are missing but planned for future iterations.

#### **Deployment Challenges:**

* + Local testing does not replicate the performance challenges that might arise in a live environment with multiple users.

### Reflection on Design Decisions

Throughout the project, several key design decisions were made:

#### **Random Forest for Diabetes Prediction:**

* + This model was chosen for its interpretability and robustness but has limitations in handling complex patterns compared to deep learning models.

#### **Neural Network for Heart Disease:**

* + While more accurate, the neural network required careful hyperparameter tuningto avoid overfitting, which increased development time.

#### **Flask as the Backend Framework:**

* + Flask provided simplicity and flexibility for this project but might face performance challenges under heavy loads, making frameworks like FastAPI or Django worth considering for future updates.

### Significance of the Work

The project demonstrates the feasibility of integrating AI into user-friendly healthcare applications. By leveraging machine learning models, it provides a cost-effective and accessible alternative to traditional diagnostic methods. While limited in scope, the system serves as a proof of concept for how technology can empower individuals in managing their health.

### Summary of Lessons Learned

#### **Iterative Development:**

* + Early feedback and incremental improvements significantly enhanced the application’s usability and performance.

#### **Importance of Data Quality:**

* + The accuracy of predictions heavily depends on the quality and diversity of training datasets.

#### **User Experience Matters:**

* + A well-designed interface improves user engagement and trust in the system.

## Conclusions and Further Work

### Summary of Achievements

The Health Check Web Application successfully integrates machine learning models to predict diabetes and heart disease in a user-friendly and responsive interface. The project demonstrates the effective use of Flask for backend operations, Bootstrap for frontend design, and machine learning libraries such as scikit-learn and TensorFlow for predictive modeling. Key achievements include:

* Development of a fully functional web application that processes user data and provides accurate health predictions.
* Implementation of a Random Forest model for diabetes prediction and a Neural Network model for heart disease prediction.
* Creation of a responsive, accessible, and aesthetically appealing user interface with features like dark mode.
* Validation of predictions with test datasets, achieving an accuracy of 78% for the diabetes model and 85% for the heart disease model.

### Design Choices and Rationale

#### **Framework Selection:**

* + Flask was chosen for its simplicity and ability to integrate Python-based machine learning models seamlessly.

#### **Machine Learning Models:**

* + Random Forest was selected for diabetes prediction due to its robustness withsmaller datasets and interpretability.
  + A Neural Network was used for heart disease prediction because of its ability to model complex relationships in the data.

#### **Frontend Design:**

* + Bootstrap was used to ensure responsiveness and a professional appearance across devices.

These choices were made to balance complexity, performance, and the project’s timeline constraints.

### Challenges and Difficulties

#### **Most Difficult Aspect:**

The integration of machine learning models with the web application was the most challenging part of the project. Handling input data inconsistencies and ensuring model predictions matched the expected format required significant effort.

#### **Why Was It Difficult?**

1. Ensuring that user inputs were correctly preprocessed to match the training data format ofthe models.
2. Debugging issues when Flask failed to communicate effectively with the serialized models due to version mismatches or data type inconsistencies.
3. Limited time for extensive hyperparameter tuning to optimize model performance.

#### **How Were These Challenges Overcome?**

1. Input validation was implemented at the backend to handle missing or incorrect data types.
2. Rigorous testing was conducted to debug and resolve Flask-model communication issues.
3. Basic hyperparameter tuning techniques were applied to improve model performance within the available time.

### Novel Contributions

#### **While the project builds on established technologies, it introduces a novel combination of:**

1. A user-friendly web interface for disease prediction, bridging the gap between research-oriented tools and end-user applications.
2. Modular architecture that allows future expansion to support additional diseases or advanced analytics.

### Lessons Learned

1. Value of Iterative Development:
   * Continuous testing and user feedback significantly improved the system’s usability and reliability.
2. Importance of Data Quality:
   * The accuracy of machine learning models heavily depends on the quality and diversity of training datasets.
3. Scalability Considerations:
   * Designing modular systems simplifies future development and expansion.

### Further Work

#### **Given more time and resources, the project could be expanded and improved in several ways:**

##### Cloud Deployment:

* + Deploy the application on a cloud platform (e.g., AWS, Heroku) to make it accessible to a wider audience and handle multiple concurrent users.

##### Additional Disease Predictions:

* + Incorporate predictive models for other conditions such as cancer, hypertension, or kidney disease.

##### Real-Time Monitoring:

* + Integrate the application with wearable devices to provide real-time health monitoring and predictions.

##### Enhanced User Features:

* + Add functionalities like user accounts, prediction history, and personalized health tips based on prediction results.

##### Improved Models:

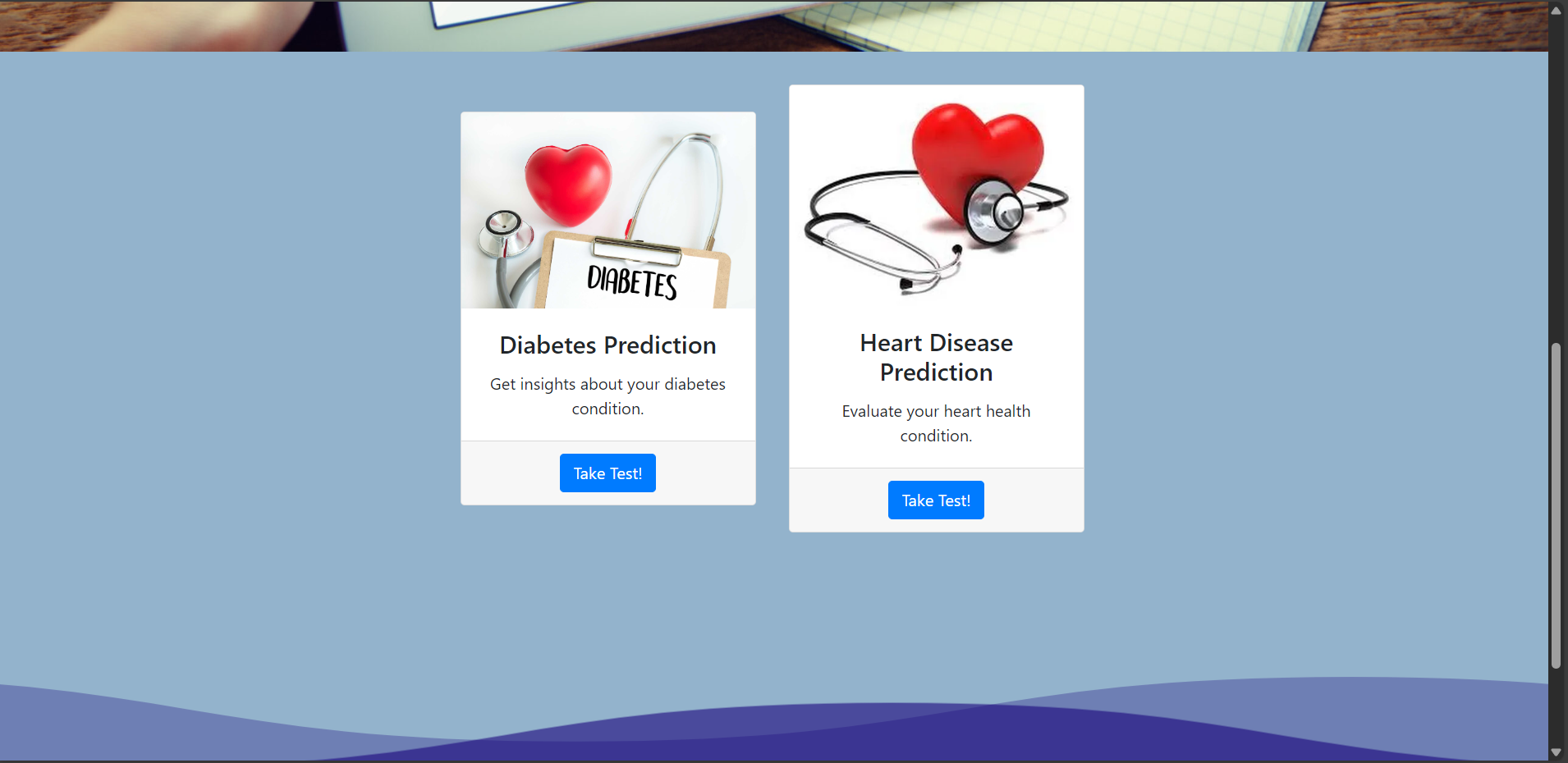
* + Use larger and more diverse datasets to retrain models for better generalizability and accuracy.

##### Advanced Visualizations:

* + Introduce interactive visualizations to help users better understand their health data and predictions.

## User Guide :

****

****

## About page:

****

## To do a Diabetes Prediction

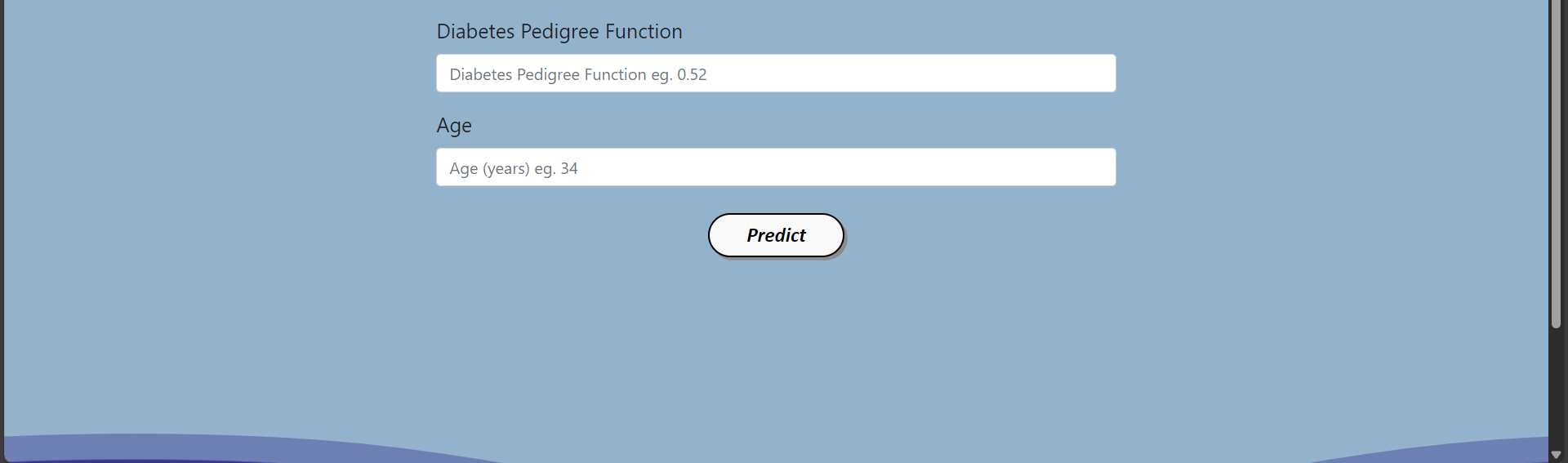
We click on Take Test

**Une image contenant texte, capture d’écran, graphisme, dessin humoristique

Description générée automatiquement**

It shows an interface like this and you need to fill it

****

****

**Example 1 of person who have diabetes:**

**Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement**

**Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement**

**The result**

**Une image contenant texte, capture d’écran, Visage humain, dessin humoristique

Description générée automatiquement**

**Example 2 of person who do not have diabetes:**

**Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement**

**Une image contenant capture d’écran, texte, logiciel, conception

Description générée automatiquement**

**The result**

**Une image contenant texte, capture d’écran, homme, Visage humain

Description générée automatiquement**

**To do a Heart Disease Prediction**

**We click on Take Test**

**Une image contenant texte, capture d’écran, graphisme, dessin humoristique

Description générée automatiquement**

**It shows an interface like this and you need to fill it**

**Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement**

**Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement**

**Example 1 of person who do not have Heart Disesase:**

**Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement**

**Une image contenant capture d’écran, texte, Rectangle, conception

Description générée automatiquement**

**The result**

**Une image contenant texte, Visage humain, capture d’écran, homme

Description générée automatiquement**

**Example 2 of person who have Heart Disesase:**

**Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement**

**Une image contenant capture d’écran, texte, Rectangle, Parallèle

Description générée automatiquement**

**The result**

**Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Description générée automatiquement**

## Bibliography

### Scikit-learn Documentation

* + **Pedregosa, F., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825-2830.**
  + **Available at: https://scikit-learn.org/stable/**

### Flask Framework Documentation

* + **Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media.**
  + **Available at: https://flask.palletsprojects.com/**

### TensorFlow Documentation

* + **Abadi, M., et al. (2016). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*.**
  + **Available at:** [**https://www.tensorflow.org/**](https://www.tensorflow.org/)

### Heart Disease Dataset

* + **Detrano, R., et al. (1989). *International application of a new probability algorithm for the diagnosis of coronary artery disease*. American Journal of Cardiology, 64(5), 304-310.**
  + **Available at:** [**https://archive.ics.uci.edu/ml/datasets/Heart+Disease**](https://archive.ics.uci.edu/ml/datasets/Heart+Disease)

### Bootstrap Documentation

* + **Bootstrap Team. (2018). *Bootstrap: Build responsive, mobile-first projects on the web with the world\u2019s most popular front-end component library*.**
  + **Available at:** [**https://getbootstrap.com/**](https://getbootstrap.com/)

### Joblib Documentation

* + **Joblib Development Team. (2023). *Joblib: Lightweight pipelining in Python*.**
  + **Available at:** [**https://joblib.readthedocs.io/**](https://joblib.readthedocs.io/)

### Kaggle for Data Hosting and Analysis

* + **Kaggle Team. (2023). *Kaggle: Your Home for Data Science*.**
  + **Available at:** [**https://www.kaggle.com/**](https://www.kaggle.com/)

## Appendices

### Test Data Sets

**The datasets used for training and testing the machine learning models are listed below. Full datasets are accessible via their respective sources:**

#### **Pima Indians Diabetes Dataset**

* + **Source: https://www.kaggle.com/uciml/pima-indians-diabetes-database**
  + **Description: Contains 768 records with 8 features relevant to diabetes prediction.**
  + **Format: CSV.**

#### **Heart Disease Dataset**

* + **Source:** [**https://archive.ics.uci.edu/ml/datasets/Heart+Disease**](https://archive.ics.uci.edu/ml/datasets/Heart+Disease)
  + **Description: Contains 303 records with 13 features related to heart disease prediction.**
  + **Format: CSV.**

**Test data examples for validation are included in the repository under /test-data.**

### Raw Results

#### **Diabetes Prediction Test Results**

| **Pregnancies** | **Glucose** | **Blood Pressure** | **Prediction** |
| --- | --- | --- | --- |
| **3** | **120** | **80** | **Negative** |
| **5** | **150** | **90** | **Positive** |

#### **Heart Disease Prediction Test Results**

| **Age** | **Cholesterol** | **Max Heart Rate** | **Prediction** |
| --- | --- | --- | --- |
| **45** | **220** | **150** | **Negative** |
| **60** | **300** | **130** | **Positive** |

### Related Resource

1. **Bootstrap Documentation:** [**https://getbootstrap.com/**](https://getbootstrap.com/)
2. **Flask Documentation: https://flask.palletsprojects.com/**
3. **TensorFlow Documentation:** [**https://www.tensorflow.org/**](https://www.tensorflow.org/)