**UNIVERSITY OF ENERGY AND NATURAL RESOURCES, SUNYANI, GHANA**

**SCHOOL OF SCIENCES**

**DEPARTMENT OF INFORMATION TECHNOLOGY AND DECISION SCIENCES**



**TITLE:**

**USING MACHINE LEARNING ALGORTHIMS TO PREDICT MALARIA AND HEPATITIS B.**

**A PROJECT WORK SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR**

**BSc INFORMATION TECHNOLOGY**

**BY:**

**EBENEZER DAMPTEY (UEB3213920)**

**DERRICK DONKOR (UEB3270122)**

**FELIX DONGBETIGR (UEB3211220)**

**AMPONSAH LUCAS (UEB3220020)**

**SUNDONG WISDOM NATONAAH (UEB3245522)**

**ASARE NANA KWASI (UEB3266622)**

**PROJECT SUPERVISOR: DR. MICHAEL OPOKU**

**SEPTEMBER, 2024**

**UNIVERSITY OF ENERGY AND NATURAL RESOURCES, SUNYANI, GHANA**

**SCHOOL OF SCIENCES**

**DEPARTMENT OF INFORMATION TECHNOLOGY AND DECISION SCIENCES**



**TITLE:**

**Disease Prediction App Using Machine Learning Algorithms to Predict Malaria and Hepatitis B.**

**A Project Work Submitted in Partial Fulfillment of The Requirements For**

**BSc INFORMATION TECHNOLOGY**

**SCHOOL OF SCIENCES**

**BY**

**EBENEZER DAMPTEY (UEB3213920)**

**DERRICK DONKOR (UEB3270122)**

**FELIX DONGBETIGR (UEB3211220)**

**AMPONSAH LUCAS (UEB3220020)**

**SUNDONG WISDOM NATONAAH (UEB3245522)**

**ASARE NANA KWASI (UEB3266622)**

**PROJECT SUPERVISOR: DR. MICHAEL OPOKU**

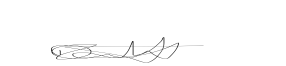
**SEPTEMBER, 2024**

# DECLARATION

DUDEWe, hereby declare that this is our own work towards the degree, and that, to the best of our knowledge, it has no material previously published by any person or entity nor material which has been presented and accepted anywhere for the award of any other degree of the University, except other works cited, which have been duly acknowledged.

EBENEZER DAMPTEY …………… 27/09/2024

(UEB3213920) Signature Date



DERRICK DONKOR …………… 27/09/2024

(UEB3270122) Signature Date



FELIX DONGBETIGR …………… 27/09/2024

(UEB3211220) Signature Date

LUCAS

AMPONSAH LUCAS …………… 27/09/2024

(UEB3220020) Signature Date

WISDOM

SUNDONG WISDOM NATONAAH …………… 27/09/2024

(UEB3245522) Signature Date

ASARE

ASARE NANA KWASI ………………. 27/09/2024

(UEB3266622) Signature Date

DR. MICHAEL OPOKU ……………… ……………..

(Supervisor) Signature Date

DR. PETER APPIAHENE ……………… ……………..

(Head of Department) Signature Date

# ACKNOWLEDGEMENT

We express our deepest gratitude for the invaluable gift of life. We extend special thanks to our parents and guardians, whose unwavering support and perseverance have guided us through our time at this university. Our heartfelt appreciation goes to our supervisor, Dr. Michael Opoku, for his generous sharing of knowledge, skillful guidance, and tireless support. His advice and mentorship have been instrumental in the success of this project.

# ABSTRACT

This project develops a Disease Prediction System using machine learning to predict malaria and hepatitis B based on patient symptoms. By analyzing patterns in symptoms and historical medical data, the system aims to improve diagnostic accuracy and support timely interventions, particularly in areas where these diseases are prevalent. The system's features include disease prediction, comprehensive descriptions, diet, precautions and workout. Utilizing advanced machine learning models such as Random Forest, Gradient Boosting Classifier, MultinomialNB and Support Vector Classifier, the system processes complex medical data to deliver reliable and personalized healthcare solutions. By addressing the limitations of traditional diagnostic methods, which can be time-consuming and resource-intensive, this project contributes to reducing the burden on healthcare providers and empowers patients with actionable insights. The implementation of this system underscores the transformative potential of AI in healthcare, particularly in enhancing the predictive diagnosis of critical diseases like malaria and hepatitis B.

# DEDICATION

This work is dedicated to our families, whose unwavering support has been our foundation. We also extend this dedication to the lecturers of the Information Technology and Decision Science department, whose guidance has been invaluable throughout our four-year journey. Lastly, we dedicate this work to our friends within the department, whose companionship and encouragement have been a constant source of strength.

# Table of Contents

[DECLARATION ii](#_Toc178368666)

[ACKNOWLEDGEMENT iii](#_Toc178368667)

[ABSTRACT iv](#_Toc178368668)

[DEDICATION v](#_Toc178368669)

[Table of Contents vi](#_Toc178368670)

[List of figures ix](#_Toc178368671)

[List of Abbreviations xi](#_Toc178368672)

[CHAPTER ONE: INTRODUCTION 1](#_Toc178368673)

[1.0 Background 1](#_Toc178368674)

[1.1 Scope of the Proposed System 2](#_Toc178368675)

[1.2 Problem Statement 3](#_Toc178368676)

[1.3 Objectives 3](#_Toc178368677)

[1.3.0 General Objective 3](#_Toc178368678)

[1.3.1 Specific Objectives 3](#_Toc178368679)

[1.4 Significance of the Project 4](#_Toc178368680)

[CHAPTER TWO: LITERATURE REVIEW 6](#_Toc178368681)

[2.0 Introduction 6](#_Toc178368682)

[2.1 Overview of Disease Prediction Systems 7](#_Toc178368683)

[2.2 Malaria Prediction Systems 9](#_Toc178368684)

[2.2.1 Machine Learning Models for Malaria Prediction 9](#_Toc178368685)

[2.2.2 Challenges in Malaria Prediction 9](#_Toc178368686)

[2.3 Hepatitis B Prediction Systems 10](#_Toc178368687)

[2.3.1 Machine Learning Approaches in Hepatitis B Prediction 10](#_Toc178368688)

[2.3.2 Limitations and Challenges 10](#_Toc178368689)

[2.4 Role of Artificial Intelligence(AI) in Disease Prediction 11](#_Toc178368690)

[2.4.1 AI in Malaria Prediction 11](#_Toc178368691)

[2.4.2 AI in Hepatitis B Prediction 11](#_Toc178368692)

[2.5 Existing Disease Prediction Systems 12](#_Toc178368693)

[CHAPTER THREE: METHODOLOGY 15](#_Toc178368694)

[3.0 Introduction 15](#_Toc178368695)

[3.1.1 Existing Systems 15](#_Toc178368696)

[3.1.2 Problems with Existing Systems 16](#_Toc178368697)

[3.1.3 Advantages of Existing Systems 17](#_Toc178368698)

[3.1.4 Disadvantages of Existing Systems 18](#_Toc178368699)

[3.2 Machine Learning Life Cycle 18](#_Toc178368700)

[3.2.1 Data Collection 19](#_Toc178368701)

[3.2.2 Data Preprocessing 20](#_Toc178368702)

[3.2.3 Model Development 22](#_Toc178368703)

[3.2.4 Model Evaluation 26](#_Toc178368704)

[3.2.5 Model Deployment 35](#_Toc178368705)

[3.3 Software Development Life Cycle 35](#_Toc178368706)

[3.3.1 Rationale Behind Waterfall Model 36](#_Toc178368707)

[3.4 Waterfall Model Life Cycle 38](#_Toc178368708)

[3.4.1 Requirements: 38](#_Toc178368709)

[3.4.2 Analysis: 39](#_Toc178368710)

[3.4.3 Design: 39](#_Toc178368711)

[3.4.4 Code (Implementation): 39](#_Toc178368712)

[3.4.5 Test: 39](#_Toc178368713)

[3.4.6 Maintenance: 40](#_Toc178368714)

[CHAPTER FOUR: DESIGN AND IMPLEMENTATION 41](#_Toc178368715)

[4.0 Introduction 41](#_Toc178368716)

[4.1 Implementation of the Malaria and Hepatitis B Prediction System 41](#_Toc178368717)

[4.1.1 Data Preprocessing 41](#_Toc178368718)

[4.1.2 Model Training 43](#_Toc178368719)

[4.1.3 Integration with the Expert System 44](#_Toc178368720)

[4.1.4 User Interface Development 45](#_Toc178368721)

[4.1.5 Model Evaluation 46](#_Toc178368722)

[4.1.6 Deployment 47](#_Toc178368723)

[4.1.7 Testing and Validation 47](#_Toc178368724)

[4.2 Challenges and Solutions 50](#_Toc178368725)

[4.3 Future Enhancements 51](#_Toc178368726)

[4.4.0 HOME SCREEN INTERFACE 52](#_Toc178368727)

[4.4.1 PREDICTION SCREEN INTERFACE 53](#_Toc178368728)

[4.4.2 CONTACT SCREEN INTERFACE 54](#_Toc178368729)

[4.4.3 ABOUT SCREEN INTERFACE 55](#_Toc178368730)

[4.4.4 Login Page 56](#_Toc178368731)

[4.4.5 Signup Screen 57](#_Toc178368732)

[CHAPTER FIVE: SUMMARY, RECOMMENDATIONS, AND CONCLUSION 58](#_Toc178368733)

[5.0 Summary 58](#_Toc178368734)

[5.1 Key Achievements 58](#_Toc178368735)

[5.2 Recommendations 59](#_Toc178368736)

[5.3 Conclusion 60](#_Toc178368737)

[REFERENCES 61](#_Toc178368738)

[APPENDICES 67](#_Toc178368739)

# List of figures

[**Figure 1 : Malaria Atlas Project (Google, n.d.) 22**](#_Toc5769)

[**Figure 2 : Machine learning life cycle. Lucidchart Content Team. (n.d.). 29**](#_Toc14947)

[**Figure 3 : Symptoms dataset 30**](#_Toc12965)

[**Figure 4 : Hyperparameter for Gradient Boosting 34**](#_Toc25482)

[**Figure 5 : Hyperparameter for SVC 35**](#_Toc32617)

[**Figure 6 : Hyperparameter for MultinomialNB 35**](#_Toc31012)

[**Figure 7 : Hyperparameter for Random Forest 35**](#_Toc5910)

[**Figure 8 : Gradient Boosting 37**](#_Toc4753)

[**Figure 9 : Multinomial NB 37**](#_Toc20519)

[**Figure 10 : Random Forest 38**](#_Toc24398)

[**Figure 11 : SVC 38**](#_Toc9256)

[**Figure 12 : Cross-Validation 39**](#_Toc25709)

[**Figure 13 : Accuracy Score for each model 40**](#_Toc30923)

[**Figure 14 : Precision for Gradient Boosting 40**](#_Toc11842)

[**Figure 15 : Precision for SVC 40**](#_Toc32569)

[**Figure 16 : Precision for Multinomial NB 41**](#_Toc19040)

[**Figure 17 : Precision for Random Forest 41**](#_Toc7590)

[**Figure 18 : Recall score 41**](#_Toc7620)

[**Figure 19 : ROC for Gradient Boosting 42**](#_Toc5928)

[**Figure 20 : ROC for Random Forest 42**](#_Toc27471)

[**Figure 21 : ROC for SVC 42**](#_Toc26057)

[**Figure 22 : F1 score 43**](#_Toc7802)

[**Figure 23 : Training Curve for Gradient Boosting 43**](#_Toc31451)

[**Figure 24 : Training Curve for Multinomial NB 44**](#_Toc11845)

[**Figure 25 : Training Curve for Random Forest 44**](#_Toc22923)

[**Figure 26 : Training Curve for SVC 44**](#_Toc11376)

[**Figure 27 : Waterfall Model Google Images. (n.d.) 48**](#_Toc16313)

[**Figure 28 : Google. (n.d.). Testing Pyramid. 60**](#_Toc21742)

[**Figure 29 : Home Screen 62**](#_Toc20743)

[**Figure 30 : Prediction Screen 63**](#_Toc12695)

[**Figure 31 : Contact Screen 64**](#_Toc30768)

[**Figure 32 : About Screen 65**](#_Toc9757)

[**Figure 33 : Login Screen 66**](#_Toc29340)

[**Figure 34 : Sign up Screen 67**](#_Toc30785)

# List of Abbreviations

* Shapley Additive exPlanations – SHAP
* Synthetic Minority Over Sampling – SMOTE
* User Acceptance Testing – UAT
* Receiver Operating Characteristic – ROC
* Area Under the Curve – AUC
* User Experience – UX
* User Interface – UI
* Support Vector Classifier – SVC
* Interquartile Range – IQR
* Software Development Life Cycle – SDLC
* True Positive – TP
* False Positive – FP
* True Negative – TN
* False Negative – FN
* Multinomial Naïve Bayes – MNB
* Existing Electronic Health Records – EHRs
* Malaria Atlas Project – MAP
* Hepatitis B Prediction Model – HBPM
* Rapid Diagnostic Tests – RDTs
* Artificial Intelligence – AI
* Machine Learning – ML

# CHAPTER ONE: INTRODUCTION

## 1.0 Background

In the modern era of technological advancements, data has become integral to decision-making processes across various industries, with the healthcare sector being a prime beneficiary. The ability to gather, analyze, and derive insights from vast amounts of medical data has revolutionized healthcare, enabling more informed decisions that enhance patient care and operational efficiency (Belsti et al., 2023). Big Data, characterized by large, diverse, and often unstructured datasets, plays a critical role in this transformation, requiring advanced analytics to identify patterns and trends that can significantly impact healthcare delivery (Tian et al., 2019).

The field of Big Data analytics in healthcare focuses on extracting valuable information from these extensive datasets, allowing for optimized operations and improved patient outcomes. For instance, platforms designed to predict diseases such as malaria and hepatitis B can utilize historical patient data, including symptom patterns and disease progression, to enhance diagnostic accuracy (Rigby et al., 2023). This is particularly crucial in areas where these diseases are prevalent, as timely and accurate diagnosis can significantly reduce morbidity and mortality rates (Figeys et al., 2023).

Malaria and hepatitis B are two diseases that continue to pose significant health challenges globally, particularly in regions with limited access to healthcare resources. Malaria, caused by the Plasmodium parasite, is a leading cause of morbidity and mortality in many tropical and subtropical regions. Hepatitis B, a viral infection that attacks the liver, can lead to chronic liver disease and is a major global health problem (Atehortúa et al., 2023). Accurate prediction and timely diagnosis of these diseases are essential for effective treatment and control, making the application of Big Data and machine learning in this context highly valuable (Dam et al., 2023).

Analyzing large volumes of medical data, including daily, monthly, and annual patient records, can provide insights into disease patterns and enable healthcare providers to anticipate outbreaks and manage resources more effectively (Chirumbolo et al., 2023). The use of machine learning models in predicting diseases like malaria and hepatitis B based on patient symptoms is a promising approach to improving healthcare delivery, particularly in resource-limited settings.

## 1.1 Scope of the Proposed System

The proposed Disease Prediction System for Malaria and Hepatitis B focuses on developing a predictive model that leverages machine learning techniques to diagnose these diseases based on patient-reported symptoms. The system aims to enhance the accuracy and timeliness of diagnoses, thereby improving patient outcomes. Specifically, the system will:

* Predict malaria and hepatitis B based on patient symptoms using machine learning models.
* Provide detailed descriptions of the predicted diseases, including possible complications.
* Recommend specific precautions to manage or prevent the disease.
* Suggest tailored diet plans and lifestyle changes suitable for the predicted disease.
* Recommend appropriate medications, including dosage and potential side effects.
* Offer suitable physical activities and exercises to aid in recovery.
* Feature a user-friendly interface for inputting symptoms and receiving disease predictions.
* Ensure the confidentiality and security of user data.

## 1.2 Problem Statement

The accurate and timely diagnosis of malaria and hepatitis B is a significant challenge for healthcare providers, especially in regions with limited medical resources. Traditional diagnostic methods, while effective, can be time-consuming and may not always capture the full spectrum of symptoms associated with these diseases. This can lead to delays in treatment, misdiagnoses, and ultimately poorer patient outcomes (Alhadreti, 2023). The sheer volume of medical data available further complicates the diagnostic process, making it difficult to identify relevant patterns and make accurate predictions (Blanco & Lourenço, 2023).

Our project seeks to address these challenges by developing a machine learning-based predictive model that analyzes historical medical data to diagnose malaria and hepatitis B accurately and promptly. By extracting key features such as symptom patterns and their correlations with these diseases, the model aims to improve diagnostic accuracy and support better healthcare outcomes.

## 1.3 Objectives

## 1.3.0 General Objective

* To develop a machine learning model that accurately predicts malaria and hepatitis B based on patient-reported symptoms.

## 1.3.1 Specific Objectives

* Assess and establish relevant data for predicting malaria and hepatitis B.
* Apply machine learning models to aid in disease prediction.
* Analyze and evaluate the performance of the predictive model.

## 1.4 Significance of the Project

The Disease Prediction System for Malaria and Hepatitis B holds significant potential for improving healthcare delivery, particularly in regions where these diseases are endemic. The system's contributions include:

* **Reducing Diagnosis Time**: By leveraging machine learning models to predict malaria and hepatitis B based on symptoms, the system can expedite the diagnostic process, leading to quicker treatment initiation and better patient outcomes.
* **Reliable Medical Recommendations**: The system provides accurate and reliable medical recommendations tailored to individual patient profiles, including symptom analysis, disease descriptions, precautions, and treatment options.
* **Supporting Informed Decisions**: Healthcare providers can benefit from the system's ability to present data-driven insights, assisting in making informed diagnostic decisions and optimizing treatment plans.
* **Patient Empowerment**: Patients gain valuable insights into their health conditions, empowering them to take proactive steps in managing their health through personalized recommendations.
* **Improving Health Outcomes**: By delivering timely and personalized medical advice, the system aims to enhance overall health outcomes, reducing the burden on healthcare facilities and improving patient well-being.
* **Enhancing Healthcare Accessibility**: Particularly in regions with limited medical resources, the system bridges gap in healthcare accessibility by providing accurate medical recommendations remotely, supporting equitable healthcare delivery.

In summary, the Disease Prediction System for Malaria and Hepatitis B represents a significant advancement in healthcare technology. By combining advanced machine learning capabilities with medical data, the system enhances diagnostic efficiency, treatment efficacy, and patient-centered care, ultimately contributing to improved health outcomes and quality of life for patients worldwide.

# CHAPTER TWO: LITERATURE REVIEW

## 2.0 Introduction

The rapid advancement of technology has profoundly impacted various sectors, with healthcare being one of the most significantly transformed. The integration of digital technologies into healthcare has led to remarkable improvements in the diagnosis, treatment, and management of diseases (Yun et al., 2023). Among the most promising technological advancements are machine learning (ML) and artificial intelligence (AI), which have introduced new possibilities for early disease detection, personalized treatment plans, and improved patient outcomes (Scholar et al., 2022).

Disease prediction systems, particularly those leveraging ML and AI, represent a critical innovation in modern healthcare. These systems analyze vast datasets, including patient symptoms, genetic information, medical history, and environmental factors, to identify patterns that may indicate the onset or progression of diseases (YUSUF & AKANDE, 2021). The predictive capabilities of these systems have the potential to revolutionize how healthcare providers approach diagnosis and treatment, shifting from reactive to proactive and preventive strategies (Tafari Shama et al., 2023).

In the context of global health, diseases such as malaria and hepatitis B continue to pose significant challenges. Malaria remains a leading cause of morbidity and mortality in many parts of the world, particularly in sub-Saharan Africa. Hepatitis B, a chronic viral infection, affects millions globally and can lead to severe liver complications, including cirrhosis and liver cancer (Lee et al., 2023). The early and accurate prediction of these diseases is crucial for effective intervention and management, reducing the burden on healthcare systems and improving patient outcomes (Rahul et al., 2023).

This chapter provides a comprehensive review of existing literature on disease prediction systems, with a particular focus on malaria and hepatitis B. It examines the current methodologies employed in these systems, including data collection, feature selection, and predictive modeling techniques. Furthermore, the chapter explores the challenges associated with disease prediction, such as data quality, model interpretability, and the integration of prediction systems into clinical practice.

Additionally, this review highlights recent advancements in the field, emphasizing the role of AI and ML in enhancing the accuracy and reliability of disease predictions. By synthesizing findings from various studies, this chapter aims to offer insights into the potential of AI-driven disease prediction systems to improve global health outcomes, particularly in resource-limited settings where these diseases are most prevalent.

## 2.1 Overview of Disease Prediction Systems

Disease prediction systems represent a significant advancement in healthcare, utilizing sophisticated algorithms to analyze extensive medical data and predict the likelihood of disease occurrence (Delanerolle et al., 2023). These systems are designed to process and interpret data from various sources, including patient symptoms, historical medical records, genetic information, lifestyle factors, and environmental influences (Guracho et al., 2023). By synthesizing this data, disease prediction systems can identify patterns and correlations that may not be immediately apparent to healthcare providers, thereby enabling earlier and more accurate diagnoses (Liu et al., 2023).

The core functionality of disease prediction systems lies in their ability to provide early warnings and risk assessments for potential health issues. This early detection capability is particularly vital in managing diseases with high morbidity and mortality rates, such as malaria and hepatitis B (Ileperuma et al., 2021). Both diseases present significant challenges to global health, particularly in regions with limited access to healthcare resources (Herlitz et al., 2023). Early and accurate prediction allows for timely intervention, which can significantly reduce the severity of the disease, improve patient outcomes, and lessen the overall burden on healthcare systems (López Bernal et al., 2023).

In the case of malaria, disease prediction systems can analyze factors such as local climate conditions, travel history, and population movement patterns, alongside patient symptoms and medical history, to predict outbreaks and individual infection risks (Hireš et al., 2023). These predictions can inform public health strategies, guiding the allocation of resources, and the implementation of preventive measures such as insecticide-treated nets and antimalarial drugs (Samadbeik et al., 2023).

Similarly, for hepatitis B, prediction systems can assess the risk of disease progression in infected individuals by analyzing a combination of viral load data, liver function tests, and genetic markers. This allows for the identification of patients at higher risk of developing serious liver conditions, such as cirrhosis or hepatocellular carcinoma, enabling earlier and more personalized treatment plans (Adamu & Singh, 2021).

Moreover, the integration of AI and machine learning into these systems has greatly enhanced their predictive accuracy and adaptability. (Ajuwon et al., 2023) Machine learning algorithms can continuously learn from new data, refining their predictions over time and adjusting to emerging trends and patterns in disease presentation. This dynamic capability ensures that disease prediction systems remain relevant and effective in rapidly changing healthcare environments (Moulaei et al., 2023).

## 2.2 Malaria Prediction Systems

Malaria is a life-threatening disease caused by Plasmodium parasites transmitted to humans through the bites of infected female Anopheles mosquitoes. Traditional diagnostic methods include microscopy and rapid diagnostic tests (RDTs), which, while effective, are time-consuming and require skilled personnel. Recent advancements in machine learning have enabled the development of predictive models that can analyze epidemiological data, climate conditions, and patient symptoms to forecast malaria outbreaks and diagnose the disease.

## 2.2.1 Machine Learning Models for Malaria Prediction

Machine learning models, such as Random Forest, Support Vector Classifier (SVC),Multinomial NB and Gradient Boosting, have been applied to predict malaria incidence. These models analyze patterns in large datasets, including climate data (temperature, humidity, rainfall), vector density, and patient symptoms, to predict the likelihood of malaria (Jovovic et al., 2023). For instance, studies have demonstrated that incorporating weather data into predictive models significantly improves the accuracy of malaria forecasts, allowing for better resource allocation and targeted interventions in high-risk areas (Mimura et al., 2023).

## 2.2.2 Challenges in Malaria Prediction

Despite the advancements, several challenges persist in developing effective malaria prediction systems. Data quality and availability remain significant hurdles, particularly in remote areas where healthcare infrastructure is lacking. Additionally, the variability in symptoms and the presence of co-infections with other diseases can complicate the prediction process. Addressing these challenges requires the development of robust models that can handle noisy and incomplete data, as well as the integration of local knowledge into predictive systems (Harabor et al., 2023).

## 2.3 Hepatitis B Prediction Systems

Hepatitis B is a viral infection that attacks the liver and can cause both acute and chronic disease. It is a major global health problem, particularly in regions with high endemicity. Early detection and treatment are critical in preventing liver complications, including cirrhosis and liver cancer. Predictive models for hepatitis B focus on analyzing patient demographics, genetic information, and clinical data to assess the risk of disease progression and the likelihood of treatment success (Valencia et al., 2023).

## 2.3.1 Machine Learning Approaches in Hepatitis B Prediction

Several machine learning approaches have been developed to predict hepatitis B infection and its progression. Logistic regression, neural networks, and ensemble methods like XGBoost have shown promise in identifying high-risk individuals and predicting disease outcomes (Marin et al., 2023). These models often utilize patient data, including viral load, liver function tests, and demographic factors, to make accurate predictions (Park et al., 2023).

## 2.3.2 Limitations and Challenges

One of the primary challenges in hepatitis B prediction is the heterogeneity of the disease, with different genotypes and variations in clinical presentation. Moreover, the chronic nature of the disease requires long-term data for accurate prediction, which may not always be available. Additionally, the integration of genetic data into predictive models poses ethical and privacy concerns that must be addressed to ensure patient confidentiality.

## 2.4 Role of Artificial Intelligence (AI) in Disease Prediction

AI and machine learning have become integral components of modern disease prediction systems. These technologies enable the analysis of large datasets, identification of patterns, and generation of insights that would be impossible for humans to achieve manually. In the context of malaria and hepatitis B, AI-driven models offer the potential to enhance diagnostic accuracy, optimize treatment strategies, and ultimately improve patient outcomes (Tariq et al., 2023).

## 2.4.1 AI in Malaria Prediction

AI models can process complex datasets, including patient records, environmental factors, and mosquito vector data, to predict malaria outbreaks with high accuracy. These models can be integrated into public health surveillance systems to provide real-time alerts and support decision-making in malaria control programs (Belsti et al., 2023).

## 2.4.2 AI in Hepatitis B Prediction

AI algorithms, particularly deep learning models, have shown promise in predicting the progression of hepatitis B by analyzing liver imaging, genetic information, and clinical data. These models can assist healthcare providers in making informed decisions about treatment plans and monitoring disease progression (Tian et al., 2019).

## 2.5 Existing Disease Prediction Systems

The application of AI and machine learning in disease prediction has led to the development of various systems that provide significant contributions to public health, particularly in predicting and managing diseases like malaria and hepatitis B.

**Malaria Atlas Project (MAP):** The Malaria Atlas Project is a leading example of how geospatial data combined with machine learning can effectively predict malaria transmission patterns globally. By integrating various data sources such as climate, population density, and historical malaria incidence, MAP offers predictive models that help in understanding the spread of malaria in different regions. (Figure 1 shows a Malaria Atlas Project). These models are crucial for public health officials in implementing targeted interventions, such as the distribution of insecticide-treated bed nets and the planning of vaccination campaigns. The real-time data processing capability of MAP allows for continuous monitoring and updating of malaria risk maps, which is essential for adaptive public health strategies.

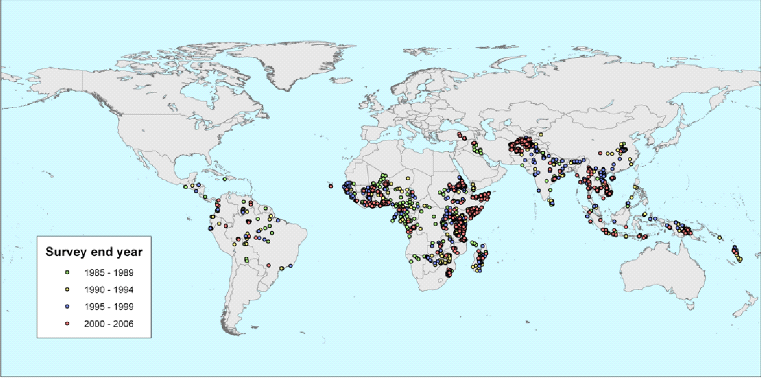


Figure : Malaria Atlas Project (Google, n.d.)

**Hepatitis B Prediction Model (HBPM):** The Hepatitis B Prediction Model leverages machine learning algorithms to predict the risk of disease progression in hepatitis B patients. This system analyzes patient data, including viral load, liver enzyme levels, and genetic markers, to provide personalized treatment recommendations. The predictive accuracy of HBPM assists healthcare providers in making informed decisions about treatment plans, such as when to initiate antiviral therapy or monitor the patient more closely. By providing a tailored approach to hepatitis B management, the HBPM has the potential to improve patient outcomes and reduce the burden of the disease on healthcare systems.

**IBM Watson for Oncology:** While not specific to malaria or hepatitis B, IBM Watson for Oncology is a prominent example of an AI-powered system that supports disease prediction and treatment planning. This system uses natural language processing and machine learning to analyze vast amounts of medical literature and patient data, offering oncologists evidence-based treatment recommendations. The adaptability of IBM Watson for Oncology serves as a model for developing similar AI-driven systems tailored to infectious diseases like malaria and hepatitis B, where real-time data analysis and personalized treatment are critical.

**Google's DeepMind Health:** Another notable example is Google's DeepMind Health, which has made significant strides in applying AI to healthcare, particularly in predictive analytics. DeepMind's algorithms have been used to predict acute kidney injury in hospital patients, demonstrating the potential of AI to foresee medical conditions before they become critical. While its primary focus is not on infectious diseases, the underlying principles of DeepMind's approach can be adapted for predicting diseases like malaria and hepatitis B, where early detection can prevent severe outcomes.

**Predictive Analytics Tools in Public Health:** Several public health organizations have adopted predictive analytics tools to anticipate disease outbreaks and trends. For instance, HealthMap integrates data from various sources, including social media, news reports, and official health data, to provide real-time surveillance of disease outbreaks worldwide. Such tools are increasingly being used to predict the spread of infectious diseases, including malaria and hepatitis B, by identifying hotspots and potential future outbreaks.

# CHAPTER THREE: METHODOLOGY

## 3.0 Introduction

This chapter presents the methodology employed in developing the Disease Prediction System for malaria and hepatitis B. It provides a detailed overview of existing prediction systems, highlights their limitations, and explains the machine learning lifecycle used to create a robust and accurate prediction model. The methodology encompasses data collection, preprocessing, model development, evaluation, and deployment phases, ensuring a systematic approach to achieving the project's objectives. Each phase has been meticulously designed to enhance the system's reliability, accuracy, and usability, ultimately aiming to improve disease diagnosis and management outcomes.

## 3.1.1 Existing Systems

Existing disease prediction systems for malaria and hepatitis B have paved the way for integrating AI and machine learning in healthcare. The Malaria Atlas Project (MAP) utilizes geospatial data and machine learning to predict malaria transmission patterns globally, while the Hepatitis B Prediction Model (HBPM) employs machine learning to assess the risk of disease progression in hepatitis B patients, offering personalized treatment recommendationsKumada, T., Toyoda, H., & Tada, T. (2023).

However, these systems often encounter challenges such as data limitations, insufficient personalization, and integration difficulties with existing healthcare infrastructure. For example, while MAP provides a global overview of malaria transmission, it may lack the granularity required for localized predictions. Similarly, HBPM might not account for individual patient variations in genetics and lifestyle. These limitations underscore the need for a more advanced approach, which this project aims to address by leveraging diverse data sources, employing rigorous preprocessing techniques, and utilizing advanced machine learning algorithms to deliver more accurate and personalized disease predictions.

## 3.1.2 Problems with Existing Systems

* **Data Limitations:** Many existing systems rely on datasets that may not fully capture the diversity of patient populations and disease manifestations. This can lead to inaccurate or incomplete predictions, particularly in regions with limited healthcare infrastructure.
* **Lack of Personalization:** Current systems often provide generalized predictions that do not consider individual patient factors such as genetic predispositions, environmental influences, and lifestyle choices. This lack of personalization can result in suboptimal disease management strategies.
* **Integration Challenges with Healthcare Infrastructure:** Integrating AI-based prediction systems with existing electronic health records (EHRs) and other healthcare IT systems can be complex. Many current solutions struggle with seamless integration, leading to fragmented data and inefficiencies in healthcare delivery.
* **Data Privacy and Security Concerns:** Handling sensitive patient data necessitates stringent security measures to prevent unauthorized access and data breaches. Existing systems often face challenges in maintaining high levels of data privacy and security, which can erode user trust and compliance with regulations.
* **Maintenance of Up-to-Date Databases:** Medical knowledge and disease patterns are constantly evolving. Existing systems often struggle to keep their databases current, which can result in outdated or inaccurate predictions. Regular updates and maintenance are essential for ensuring the reliability of these systems.
* **User Interface and Experience:** Many existing systems lack user-friendly interfaces, making it difficult for healthcare providers and patients to effectively interact with the system. A poorly designed interface can hinder the system's adoption and usability.
* **Bias in AI Algorithms:** AI algorithms can inadvertently incorporate biases from the training data, leading to biased predictions. Ensuring fairness and impartiality in disease prediction is a significant challenge that existing systems often struggle to address.

## 3.1.3 Advantages of Existing Systems

* **Enhanced Diagnostic Support:** By leveraging AI and machine learning, existing systems can analyze large volumes of medical data, aiding healthcare providers in diagnosing diseases more accurately and efficiently.
* **Geospatial Analysis for Malaria:** The Malaria Atlas Project provides valuable insights into malaria transmission patterns, which can inform public health interventions and resource allocation.
* **Personalized Treatment for Hepatitis B:** The Hepatitis B Prediction Model offers personalized treatment recommendations based on an individual’s risk profile, enhancing patient care and outcomes.

## 3.1.4 Disadvantages of Existing Systems

* **Limited Accuracy Due to Insufficient Data:** The effectiveness of these systems is often compromised by the limited scope and quality of available data, resulting in less accurate predictions.
* **Lack of Personalization:** Many existing systems fail to account for individual patient differences, leading to generalized predictions that may not be optimal for specific cases.
* **Integration Challenges:** Existing systems often face difficulties in integrating with healthcare IT infrastructure, leading to fragmented workflows and inefficiencies.
* **Data Privacy and Security Issues:** Protecting sensitive patient information is a significant challenge. Many current systems struggle with implementing and maintaining robust data security measures, potentially leading to privacy breaches.
* **Maintenance and Update Requirements:** Keeping the databases of these systems up-to-date with the latest medical knowledge is a continuous and resource-intensive task, often neglected in existing systems.
* **User Interface Challenges:** The usability of existing systems can be hindered by complex or poorly designed interfaces, making it difficult for users to interact effectively with the system.

## 3.2 Machine Learning Life Cycle

The development of the Disease Prediction System for malaria and hepatitis B adheres to a structured machine learning lifecycle. This lifecycle ensures that each phase, from data gathering to model deployment, is systematically executed to achieve high levels of accuracy, reliability, and efficiency. (Figure 2 is an outline of each phase within the machine learning lifecycle).

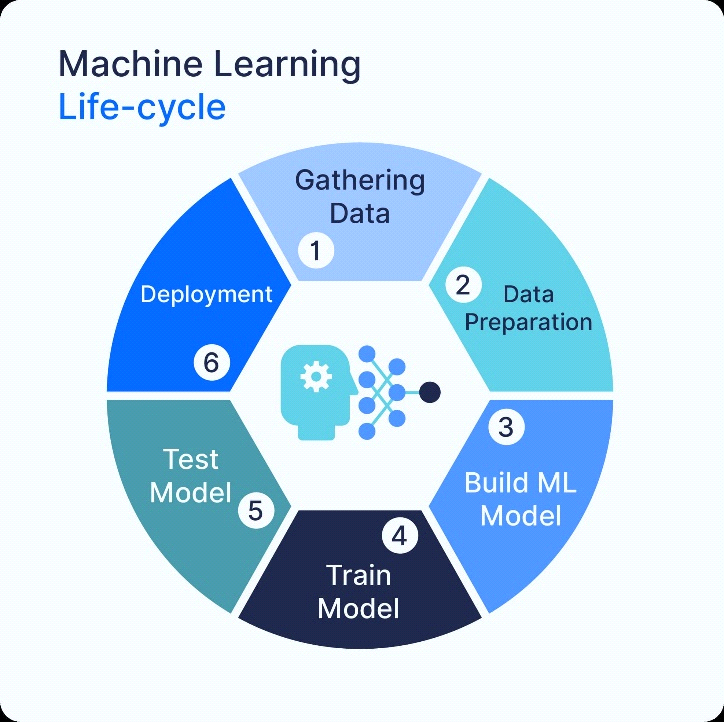


Figure : Machine learning life cycle. Lucidchart Content Team. (n.d.).

## 3.2.1 Data Collection

Data collection is the foundational phase of the machine learning lifecycle, and the success of the disease prediction system depends on the quality and comprehensiveness of the data gathered. For this project, datasets related to malaria and hepatitis B were carefully sourced to ensure robust model development. (Figure 3 shows a dataset of Symptoms). The primary dataset used in this research includes Training.csv, which contains the training data for building the predictive model; Symptoms.csv, which lists symptoms associated with various diseases; and Disease\_description.csv, which provides detailed descriptions of these diseases.

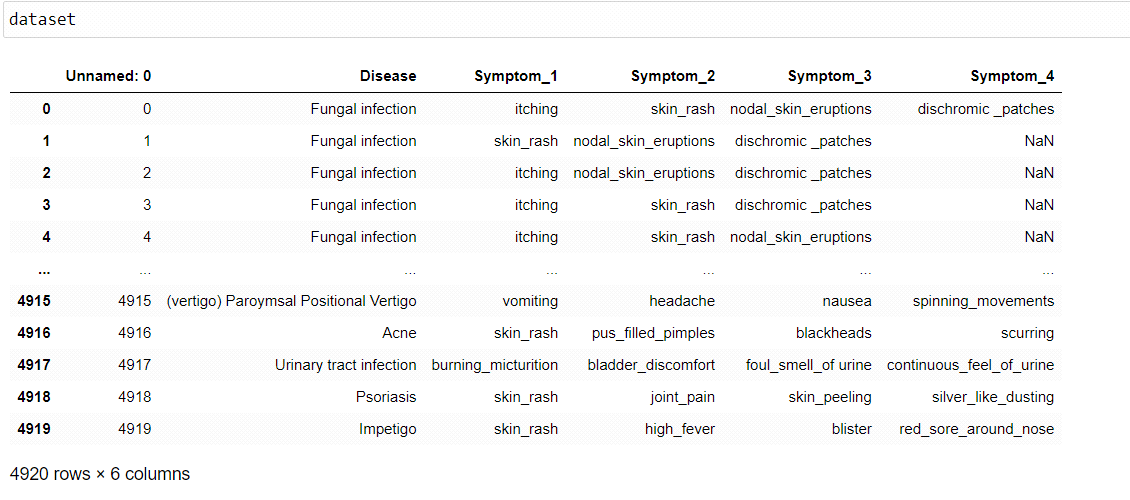


Figure : Symptoms dataset

## 3.2.2 Data Preprocessing

Data preprocessing is a crucial step in the data science pipeline, as it lays the foundation for accurate and meaningful analysis, modeling, and decision-making. After the data collection phase, several preprocessing steps were applied to ensure the dataset's quality and suitability for building an effective disease prediction model. The following steps were performed:

**Data Cleaning**

Handling missing values, outliers, and inconsistencies in the dataset is essential to ensure accurate and reliable analysis and modeling. In this project:

* **Duplicate Removal:** The dataset initially contained some duplicate entries. To maintain the integrity of the analysis, these duplicates were identified and removed using the drop\_duplicates () method in pandas.
* **Filtering Specific Diseases:** The dataset was filtered to focus specifically on the diseases of interest, namely malaria and hepatitis B. This was achieved by retaining only the rows where the prognosis column matched these diseases.

No missing values were reported in the provided dataset; hence, no imputation or further cleaning was necessary in this step.

**Data Transformation**

Transformations were applied to the data to prepare it for model training:

* **Feature Selection:** A set of relevant symptoms was selected based on domain knowledge. These symptoms were identified as key indicators for malaria and hepatitis B and were used to filter the dataset. This step reduced dimensionality and focused the analysis on the most impactful variables.
* **Encoding Categorical Variables:** The prognosis column, representing the disease labels, was encoded using LabelEncoder from scikit-learn. This transformation converted the categorical labels into numerical format, which is required for model training.

**Normalization Using Z-Score**

Data normalization was performed to standardize the feature values, making them comparable across different scales:

* **Z-Score Standardization:** For models like Support Vector Classifier (SVC), Random Forest, and Gradient Boosting, the symptom data was normalized using Standard Scaler. This technique transforms the data so that each feature has a mean of 0 and a standard deviation of 1, ensuring that all variables contribute equally to the model's decision-making process.

## 3.2.3 Model Development

The model development phase is a critical component of the machine learning pipeline, where various machine learning algorithms are selected, trained, and fine-tuned to build a robust predictive model. The goal is to identify the best model or combination of models that can accurately predict diseases such as malaria and hepatitis B based on symptoms. The following steps were undertaken during this phase:

**Algorithm Selection**

Several machine learning algorithms were selected based on their ability to handle complex relationships and non-linearities, which are often present in medical data. The algorithms considered in this project include:

* **Random Forest:** A versatile and powerful ensemble method that constructs multiple decision trees and merges them to obtain a more accurate and stable prediction. It is particularly useful for its robustness to overfitting and its ability to handle high-dimensional data.
* **Gradient Boosting:** Another ensemble method that builds models sequentially, with each new model correcting errors made by the previous ones. It is effective in minimizing errors and improving prediction accuracy, especially for imbalanced datasets.
* **Support Vector Classifier (SVC):** A robust classification algorithm that works well for both linear and non-linear data. SVC is particularly effective in high-dimensional spaces and is used for cases where the number of dimensions exceeds the number of samples.
* **Multinomial Naive Bayes (MNB):** Although simple, this algorithm is effective for certain types of problems, especially when dealing with categorical data or data that follows a multinomial distribution. In this project, MNB was included as a baseline model to compare its performance against more complex models.

**Model Training**

The selected models were trained using the preprocessed dataset, where the models learned to identify patterns and relationships between the input features (symptoms) and the output labels (prognosis for malaria or hepatitis B).

* **Training Process:** Each algorithm was trained on the dataset using a stratified train-test split to ensure that both training and testing sets were representative of the overall data distribution. The training phase involved fitting the models to the data so that they could learn to predict the disease based on the symptoms provided.
* **Feature Importance:** For models like Random Forest, feature importance was evaluated to understand which symptoms contributed most significantly to the predictions.

best\_model = RandomForestClassifier (n\_estimators=100, random\_state=42) best\_model.fit (X\_train\_filtered, y\_train\_filtered) feature\_importances = best\_model. feature\_importances\_

**Hyperparameters Tuning**

To optimize model performance, hyperparameters that control the behavior of the algorithms were fine-tuned. (Figures 4,5,6 and 7 shows the hyperparameter results for Gradient Boosting, SVC, Multinomial NB and Random Forest respectively) This was accomplished using the following techniques:

* **Grid Search:** A comprehensive search over a predefined set of hyperparameters was performed for each model. Grid search systematically works through multiple combinations of parameter values, cross-validating as it goes to determine the best combination of hyperparameters for each model.
* **Cross-Validation:** Cross-validation was employed during the grid search to evaluate the model's performance on different subsets of the training data. This technique helps to prevent overfitting and ensures that the model generalizes well to unseen data.
* **Performance Metrics:** Model performance was evaluated using metrics such as accuracy, confusion matrix, and classification report to ensure that the models were performing well on both the training and testing data.

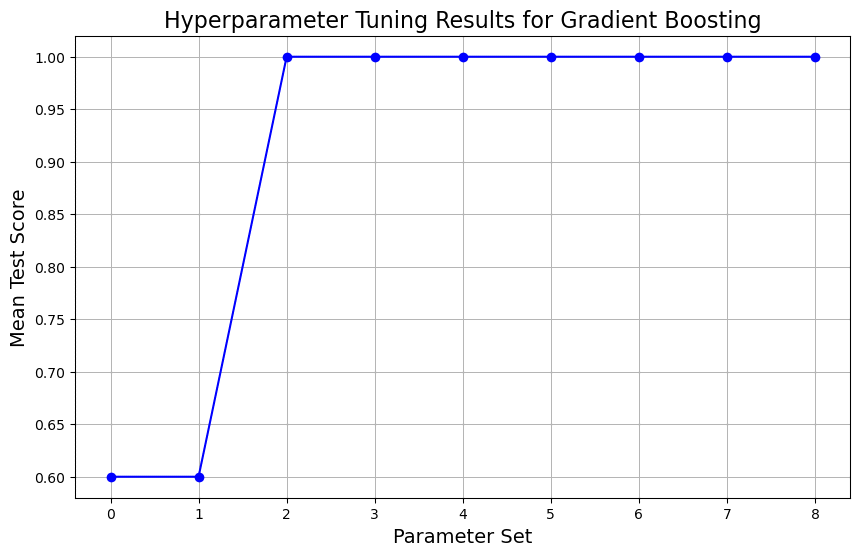


Figure : Hyperparameter for Gradient Boosting

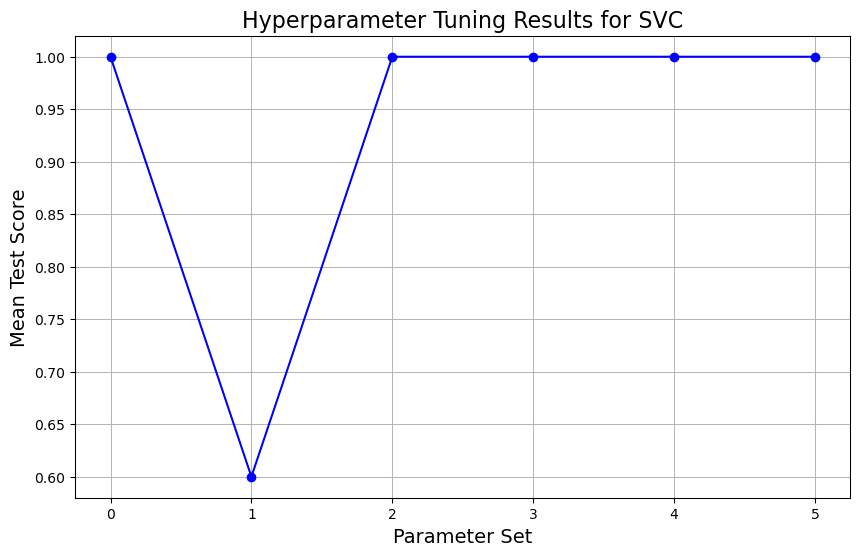


Figure : Hyperparameter for SVC

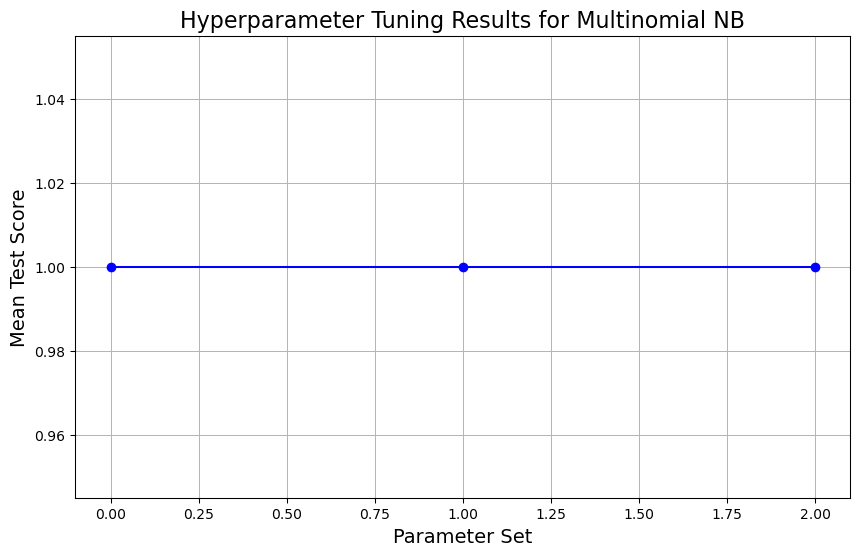


Figure : Hyperparameter for MultinomialNB

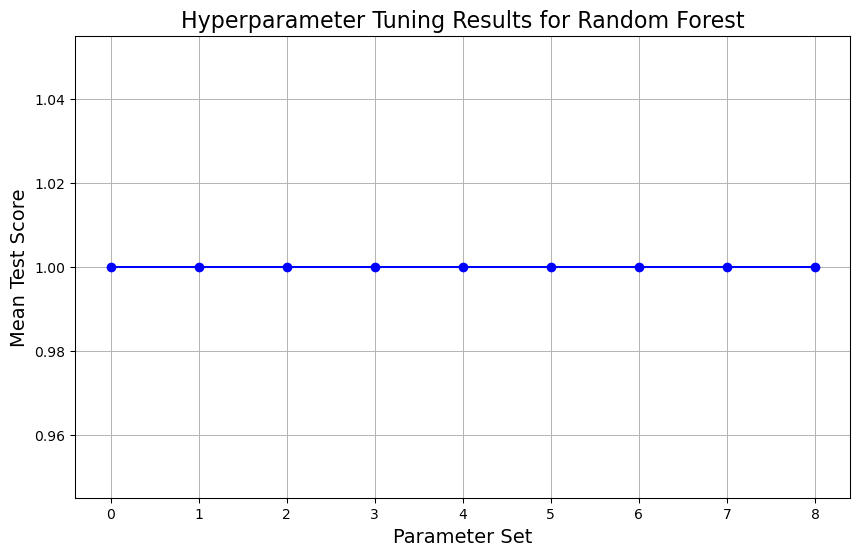


Figure : Hyperparameter for Random Forest

**Ensemble Methods**

To enhance predictive accuracy and build a more robust model, ensemble methods were employed:

* **Stacking, Bagging, and Boosting:** These ensemble techniques combine the predictions of multiple models to produce a more accurate and reliable prediction. In this project, ensemble methods were particularly useful in leveraging the strengths of different models, such as Random Forest and Gradient Boosting, to improve overall prediction accuracy.
* **Model Performance Comparison:** The performance of individual models and ensemble methods was compared to determine the best-performing approach. The results were visualized using accuracy scores, confusion matrices, and ROC curves to illustrate the effectiveness of each model and the improvements gained through resembling.

## 3.2.4 Model Evaluation

After training the models, an essential step was to evaluate their performance to ensure they met the desired accuracy and reliability standards. Several evaluation techniques were employed to comprehensively assess the models' effectiveness in predicting diseases like malaria and hepatitis B. The following methods were used:

**Confusion Matrix**

The confusion matrix is a crucial tool for evaluating classification models, providing a detailed breakdown of the model's performance by showing the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). (Figures 8, 9, 10 and 11 shows the Confusion Matrix for Gradient Boosting, Multinomial NB, Random Forest and SVC respectively) This matrix helps in understanding where the model is making errors and the types of errors it is prone to (e.g., more false positives or false negatives).

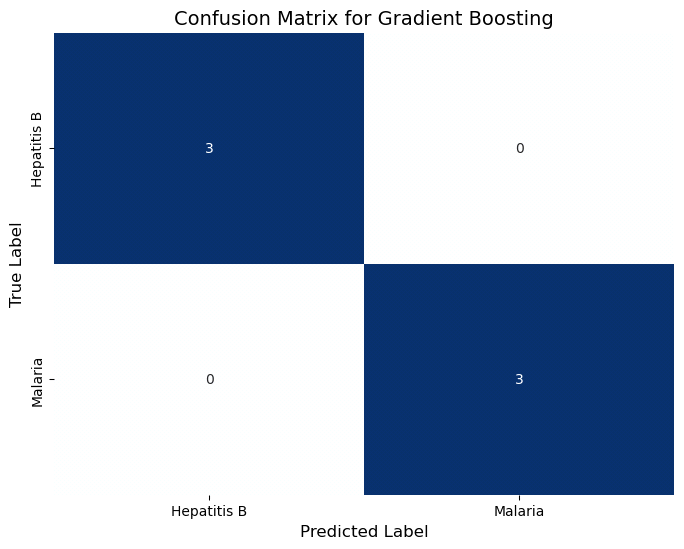


Figure : Gradient Boosting

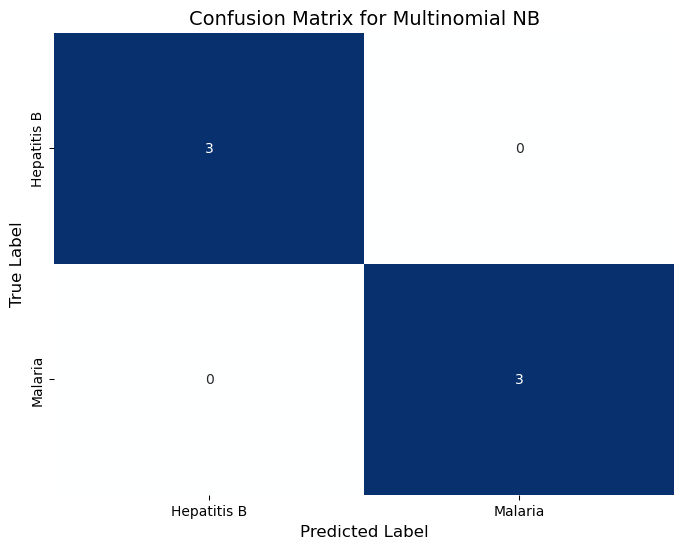


Figure : Multinomial NB

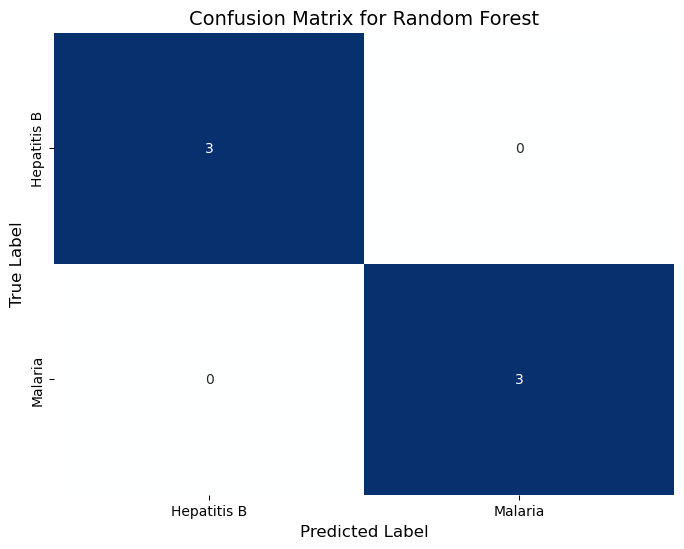


Figure : Random Forest

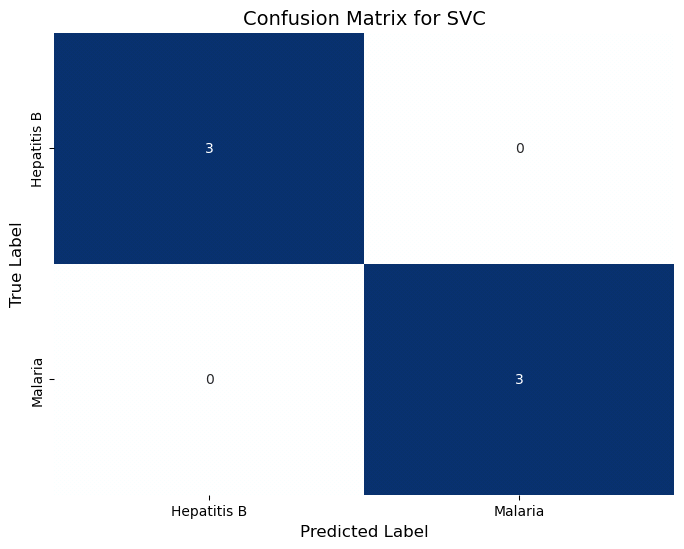


Figure : SVC

**Cross-Validation**

To ensure the model's consistency and generalizability across different subsets of data, K-fold cross-validation was employed. (Figure 12 shows a Cross validation) This technique divides the dataset into K subsets, trains the model on K-1 subsets, and tests it on the remaining subset, rotating this process K times. The results are then averaged to provide a more robust estimate of the model’s performance.

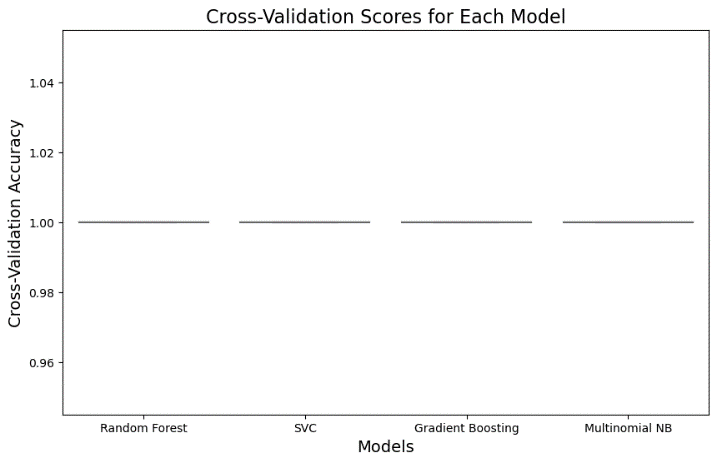


Figure : Cross-Validation

**Accuracy, Precision, and Recall**

These metrics were calculated to assess the model’s overall correctness, its ability to identify true positive cases, and its effectiveness in minimizing false negatives:

* **Accuracy:** Measures the proportion of correct predictions (both TP and TN) out of the total number of predictions. (Figure 13 shows the Accuracy score for each model)

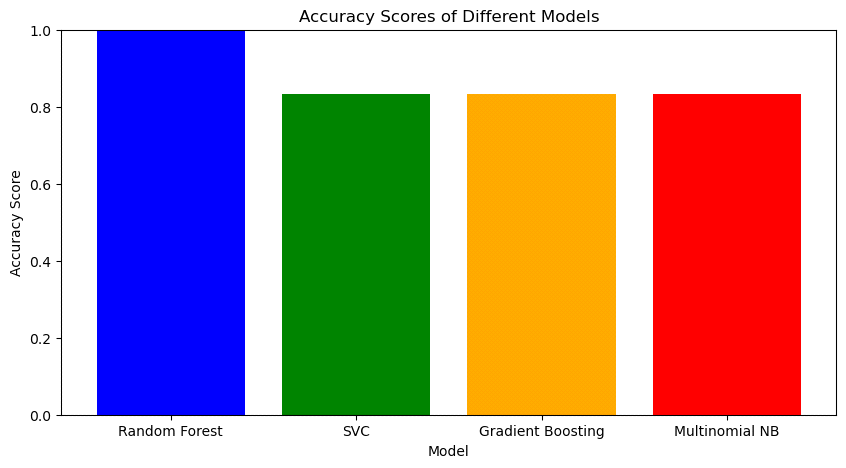


Figure : Accuracy Score for each model

* **Precision:** Indicates the proportion of true positive cases among all positive predictions, helping to understand the model's effectiveness in avoiding false positives. (Figures 14, 15, 16 and 17 shows the precision curve for Gradient Boosting, SVC, Multinomial NB and Random Forest respectively).

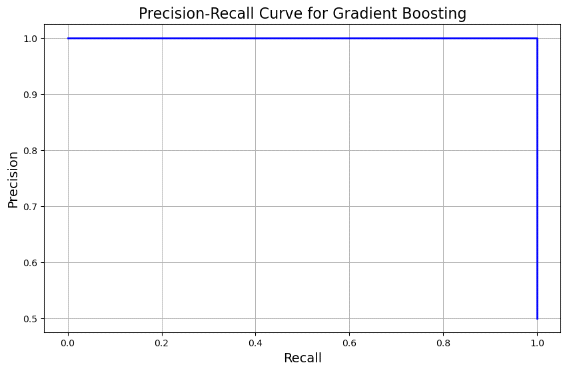


Figure : Precision for Gradient Boosting

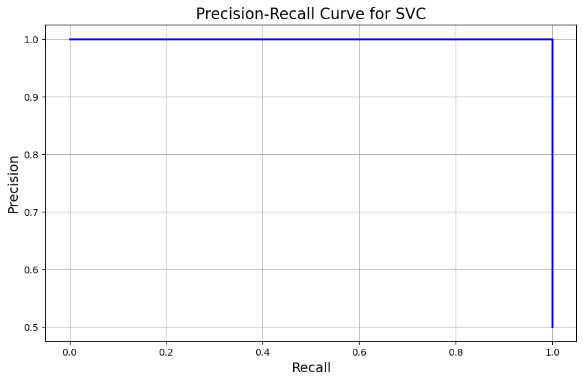


Figure : Precision for SVC

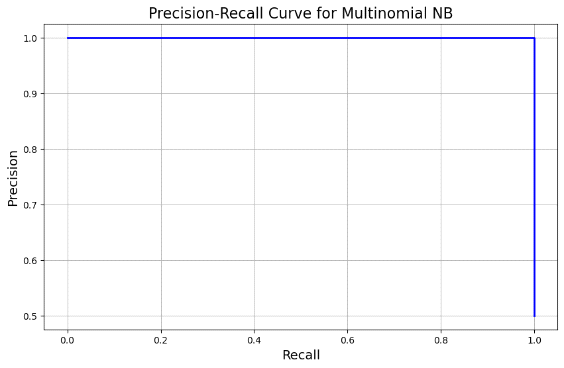


Figure : Precision for Multinomial NB

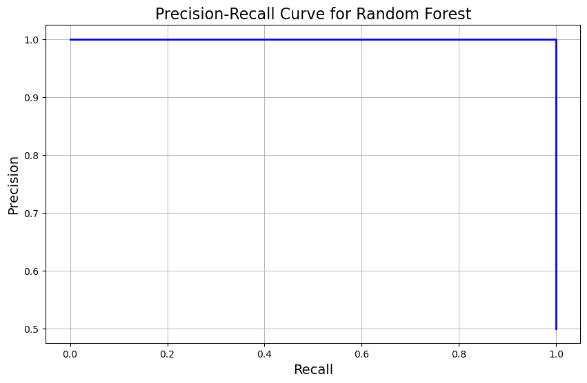


Figure : Precision for Random Forest

* **Recall (Sensitivity):** Measures the model’s ability to correctly identify all true positive cases, thereby reducing false negatives. (Figure 18 shows the Recall score)

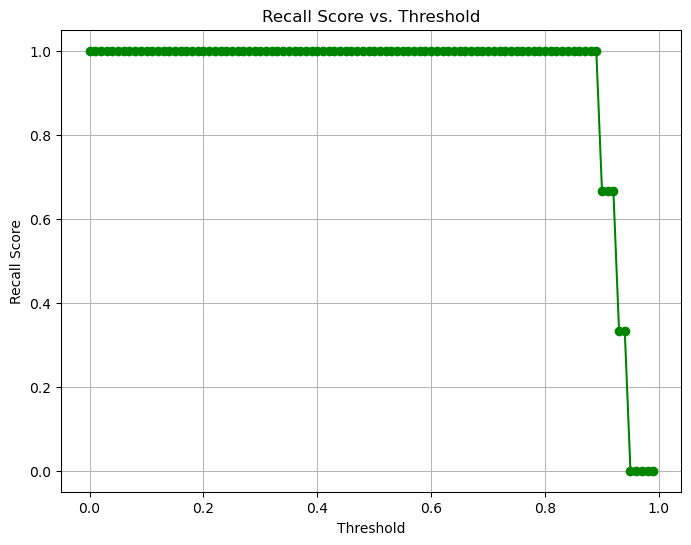


Figure : Recall score

**Receiver Operating Characteristic (ROC) Curve**

The ROC curve was plotted to visualize the model's ability to distinguish between positive and negative cases across various threshold settings. (Figure 19, 20 and 21 shows the ROC curve for Gradient Boosting, Random Forest and SVC respectively).

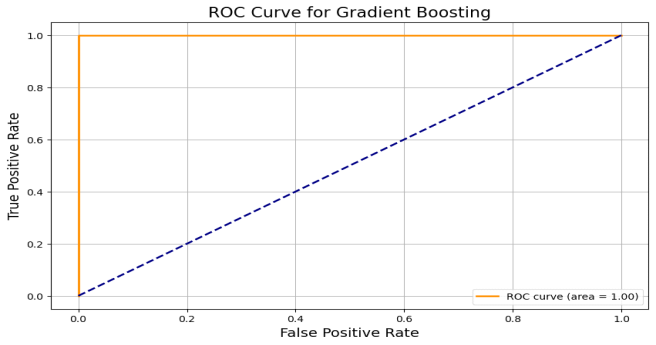


Figure : ROC for Gradient Boosting

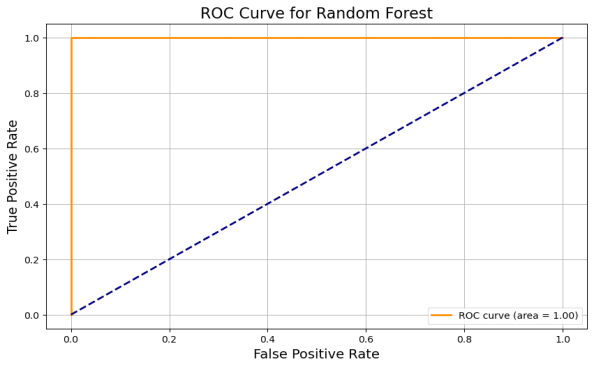


Figure : ROC for Random Forest

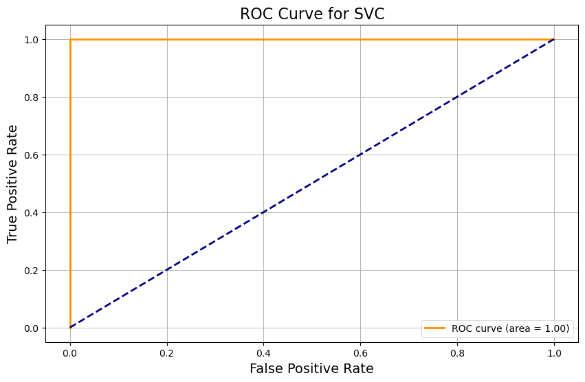


Figure : ROC for SVC

**F1 Score**

The F1 score, which balances precision and recall, was computed to evaluate the model's effectiveness, especially in handling imbalanced datasets. (Figure 22 shows the F1 score). The F1 score is particularly important when the cost of false positives and false negatives is significant.

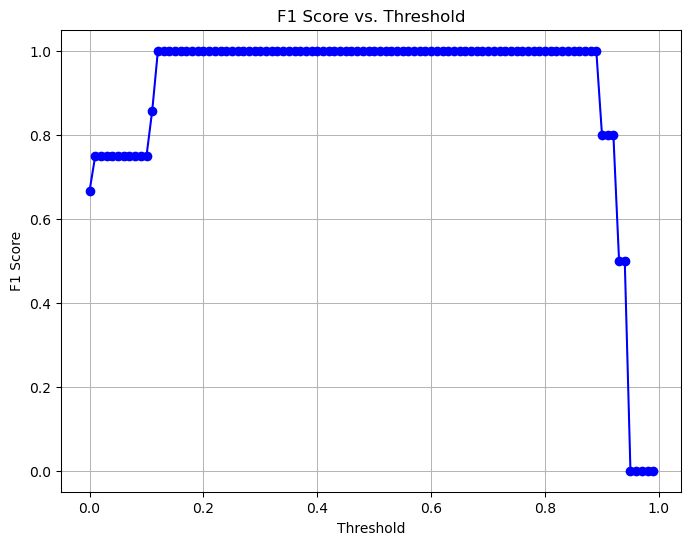


Figure : F1 score

**Training learning Curve**

A training learning curve plots the performance of a model on the training dataset as a function of the number of training iterations or epochs. (Figures 23, 24, 25 and 26 shows the Training Curve for Gradient Boosting, Multinomial NB, Random Forest and SVC respectively) It helps in understanding how well the model is learning the training data and can indicate whether more training is beneficial or if the model is already well-trained.

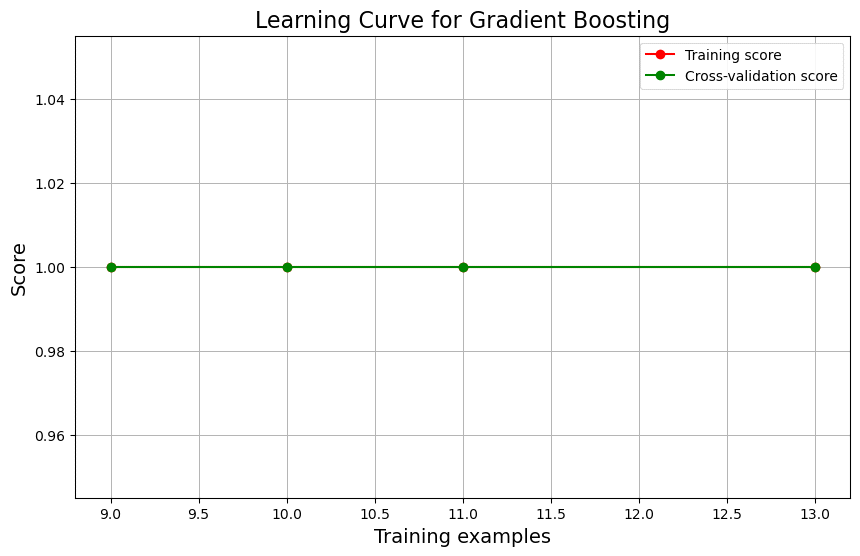


Figure : Training Curve for Gradient Boosting

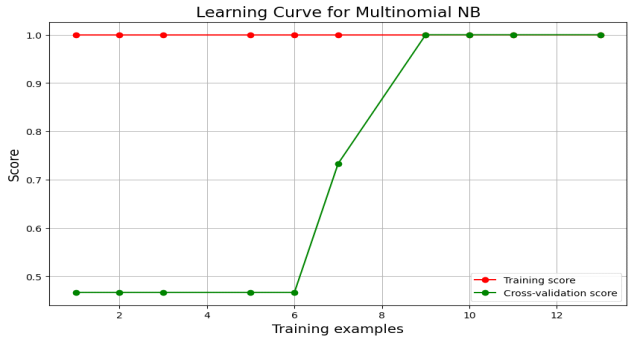


Figure : Training Curve for Multinomial NB



Figure : Training Curve for Random Forest

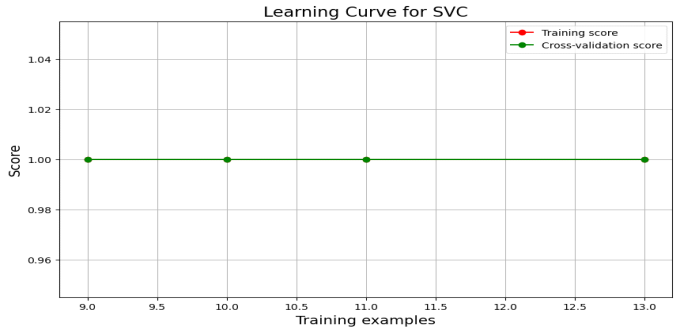


Figure : Training Curve for SVC

## 3.2.5 Model Deployment

Following rigorous evaluation, the best-performing models were selected for deployment. The deployment phase involved the following steps:

* **Integration with Frontend:** The trained model was integrated with the user interface, allowing healthcare providers or patients to input symptoms and receive predictions about the likelihood of malaria or hepatitis B.
* **API Development:** A RESTful API was developed to enable communication between the frontend application and the backend model, facilitating seamless interaction and data exchange.
* **Model Monitoring:** Post-deployment, the model was continuously monitored to ensure consistent performance in a real-world setting, tracking prediction accuracy and making adjustments as needed.

## 3.3 Software Development Life Cycle

The Software Development Life Cycle (SDLC) is a structured process used by software developers to design, develop, test, and deploy software. It provides a framework for managing the phases involved in the creation of software, ensuring that the final product meets the intended requirements and is of high quality.

## 3.3.1 Rationale Behind Waterfall Model

The Waterfall Model is a traditional and linear approach to software development that emphasizes a structured and sequential progression through distinct phases. (Figure 27 shows the life cycle of a Waterfall model) The rationale behind this model lies in its ability to provide a clear, organized, and systematic framework, which is particularly effective for projects with well-defined requirements and minimal expected changes.

* **Clear and Structured Process:**
* The Waterfall Model offers a straightforward and easy-to-understand process, where each phase has a distinct purpose and is completed before the next phase begins. This clear structure helps in organizing and managing large projects by breaking them down into manageable stages.
* **Defined Requirements and Predictability:**
* Waterfall is best suited for projects with well-defined and stable requirements that are unlikely to change. Since all requirements are gathered and documented upfront, there is a predictable outcome, making it easier to estimate costs, timelines, and resource needs.
* **Phase-by-Phase Progression:**
* The sequential nature of the Waterfall Model ensures that each phase is thoroughly completed and reviewed before moving to the next, minimizing the chances of errors propagating through the development process. This phase-by-phase progression fosters discipline and ensures that all aspects of the project are addressed systematically.
* **Ease of Management and Documentation:**
* The Waterfall Model emphasizes thorough documentation at each stage of the development process, which is beneficial for maintaining project clarity and traceability. This comprehensive documentation aids in better project management and facilitates communication among stakeholders.
* **Quality Assurance through Rigorous Testing:**
* Testing is a dedicated phase in the Waterfall Model, occurring only after the development phase is complete. This allows for a focused and rigorous testing process, ensuring that the system is thoroughly evaluated against the initial requirements.
* **Stability and Control:**
* The Waterfall Model is ideal for projects where stability and control are paramount. Its linear and systematic approach provides a controlled environment, reducing the likelihood of scope creep and ensuring that changes are minimized once the project is underway.
* **Suited for Projects with Fixed Deadlines:**
* Because of its predictability and structured nature, the Waterfall Model is well-suited for projects with fixed deadlines and deliverables. This makes it an ideal choice for projects where time constraints and budget considerations are critical.

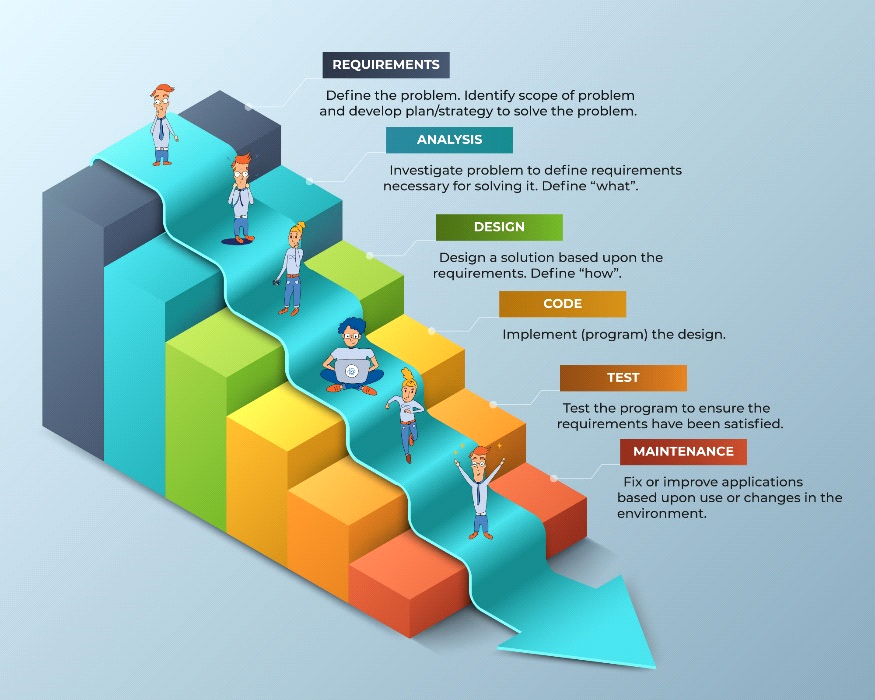


Figure : Waterfall Model Google Images. (n.d.)

## 3.4 Waterfall Model Life Cycle

## 3.4.1 Requirements:

* **Description:** This phase involves defining the problem that the software will solve. It includes identifying the scope of the problem and developing a plan or strategy to address it.
* **Purpose:** To gather and document all the necessary requirements from stakeholders to ensure a clear understanding of what the software needs to achieve.

## 3.4.2 Analysis:

* **Description:** In this phase, the problem is further investigated to define the specific requirements needed for solving it. This step answers the question of "what" the system will do.
* **Purpose:** To translate the gathered requirements into detailed functional specifications.

## 3.4.3 Design:

* **Description:** Based on the requirements and analysis, a solution is designed. This phase defines "how" the system will meet the requirements.
* **Purpose:** To create a blueprint for building the software, including the architecture, interface design, and data models.

## 3.4.4 Code (Implementation):

* **Description:** The actual coding and development of the software take place here. The design is translated into a working software product by implementing the code.
* **Purpose:** To build the software components as per the design specifications.

## 3.4.5 Test:

* **Description:** The developed software is rigorously tested to ensure it meets the requirements and is free of defects. Testing is done to validate that the program works as expected.
* **Purpose:** To identify and fix any issues or bugs before the software is deployed.

## 3.4.6 Maintenance:

* **Description:** After deployment, the software enters the maintenance phase. This involves fixing any issues that arise during use and making updates or improvements as needed.
* **Purpose:** To ensure the software remains functional, relevant, and up-to-date over time.

# CHAPTER FOUR: DESIGN AND IMPLEMENTATION

## 4.0 Introduction

In this chapter, we delve into the practical design and implementation of the Malaria and Hepatitis B prediction system, detailing the steps taken to develop and integrate the predictive models into the expert system. We also present the results of our model evaluation and discuss the implications of our findings. The system was built using the following technological stack: Python, Flask, machine learning libraries, and Flutter. Specifically, we employed the following machine learning algorithms: Random Forest, Support Vector Classifier, Gradient Boosting Classifier, and Multinomial Naive Bayes.

This chapter not only outlines the steps taken in developing and integrating the predictive models with the expert system but also unveils the outcomes of our model evaluation, providing a comprehensive overview of the system's capabilities and performance.

## 4.1 Implementation of the Malaria and Hepatitis B Prediction System

The implementation of the Malaria and Hepatitis B prediction system involved several key steps, including data preprocessing, model training, integration with the expert system, and user interface development.

## 4.1.1 Data Preprocessing

Data preprocessing is a foundational step in any machine learning project, particularly when dealing with healthcare data where accuracy is paramount. The datasets used for predicting Malaria and Hepatitis B were obtained from reputable sources and contained various attributes such as patient age, gender, symptoms, medical history, and other relevant features.

* **Data Cleaning:** The raw data contained missing values, outliers, and inconsistencies that could potentially degrade model performance. Missing values were handled using techniques such as mean imputation for continuous variables and mode imputation for categorical variables. Outliers were detected using statistical methods like the Interquartile Range (IQR) and were either corrected or removed depending on their impact on the model.
* **Feature Engineering:** Feature engineering was performed to create new features from existing data that could help improve model performance. For example, the duration of symptoms before seeking medical help was calculated and added as a new feature. Additionally, categorical variables such as gender and medical history were encoded using one-hot encoding to make them suitable for machine learning algorithms.
* **Data Normalization and Scaling:** Since different features in the dataset were measured on different scales, normalization and scaling were necessary to ensure that all features contributed equally to the model training process. Techniques such as Min-Max scaling and Standardization were employed to transform the data into a consistent range, which is especially important for distance-based algorithms like K-Nearest Neighbors.
* **Dataset Splitting:** The preprocessed data was then split into training, validation, and test sets. The training set was used to train the models, the validation set to tune hyperparameters and prevent overfitting, and the test set to evaluate the final performance of the models. A typical split ratio of 70:15:15 was used, ensuring that each set was representative of the overall dataset.

## 4.1.2 Model Training

Model training involved selecting appropriate machine learning algorithms, tuning them to achieve the best performance, and then training them on the prepared data. The following algorithms were chosen based on their suitability for classification tasks in medical data:

* **Random Forest Classifier:** This ensemble learning method was chosen for its ability to handle large datasets with high dimensionality and its robustness to overfitting. The model was trained using a large number of decision trees, with each tree being trained on a random subset of the data. The final prediction was made by aggregating the predictions of all individual trees (majority voting). Hyperparameters such as the number of trees (n\_estimators), maximum depth (max\_depth), and the number of features considered for splitting (max\_features) were tuned using grid search and cross-validation.
* **Support Vector Classifier (SVC):** SVC was selected for its effectiveness in high-dimensional spaces and its ability to model complex decision boundaries. The model was trained using a Radial Basis Function (RBF) kernel, which is well-suited for non-linear classification problems. Key hyperparameters like the regularization parameter (C) and kernel coefficient (gamma) were optimized to balance the trade-off between maximizing the margin and minimizing classification errors.
* **Gradient Boosting Classifier:** This model was chosen for its ability to build models sequentially, each one correcting the errors of its predecessor. It is particularly effective in reducing bias and variance, making it ideal for medical prediction tasks. Hyperparameters such as the learning rate, the number of boosting stages (n\_estimators), and the maximum depth of individual trees were carefully selected to prevent overfitting while ensuring high accuracy.
* **Multinomial Naive Bayes:** This algorithm, based on Bayes' theorem, was selected for its simplicity and efficiency, especially with categorical data. It was particularly useful for modeling the probability of different diagnoses based on symptom presence. The model assumes that the features are conditionally independent given the class label, an assumption that, while simplistic, provides strong performance in practice. The smoothing parameter (alpha) was fine-tuned to handle cases with zero probability estimates.
* **Model Optimization:** After initial training, model optimization was performed using techniques such as Hyperparameter tuning, cross-validation, and ensemble methods. Ensemble methods like bagging and boosting were used to combine the strengths of different models, thereby improving overall prediction accuracy. Cross-validation was particularly important for assessing the models' ability to generalize to unseen data, helping to avoid overfitting.

## 4.1.3 Integration with the Expert System

The integration of the predictive models with the expert system was a crucial step in ensuring that the system could provide real-time, accurate predictions to healthcare professionals and patients.

* **Architecture Design:** The expert system was designed with a modular architecture consisting of three main components: the knowledge base, the inference engine, and the user interface. The predictive models were integrated into the knowledge base, allowing the inference engine to leverage the models' outputs when making diagnostic decisions.
* **RESTful API Development:** A RESTful API was developed using Flask to enable communication between the predictive models and the expert system. The API endpoints were designed to accept patient data, process it through the predictive models, and return the likelihood of Malaria or Hepatitis B, along with recommended actions. The API was secured using authentication tokens to ensure that only authorized users could access the prediction services.
* **Error Handling and Logging:** Robust error handling mechanisms were implemented to ensure that any issues during the prediction process were captured and logged. This included handling exceptions related to data input, model execution, and API communication. Logs were stored in a centralized logging system, providing valuable insights for troubleshooting and continuous improvement of the system.

## 4.1.4 User Interface Development

The user interface (UI) was developed using Flutter, providing a cross-platform solution that ensures a consistent user experience across different devices. The UI was designed to be both intuitive and responsive, catering to users with varying levels of technical expertise.

* **Design Principles:** The UI was built with a focus on simplicity and ease of use. Key design principles included a clean layout, clear navigation, and minimalistic design elements that prioritize functionality. The interface was tested with potential users, including healthcare professionals, to gather feedback and ensure that it met their needs.
* **Features:** The UI includes several key features such as input forms for entering patient data, a dashboard for viewing prediction results, and a section for accessing historical data and reports. The prediction results are displayed in a user-friendly format, with visual aids like charts and graphs to help users interpret the data quickly.
* **Responsive Design:** The UI was designed to be fully responsive, ensuring that it adapts to different screen sizes and orientations. This was particularly important for ensuring that healthcare providers could access the system on tablets and mobile devices while in the field.
* **Internationalization:** The UI was also designed with internationalization in mind, allowing for the system to be easily adapted to different languages and regions. This ensures that the system can be used in diverse geographical locations, making it a valuable tool in global health initiatives.
* **User Experience (UX):** The user experience was enhanced through thoughtful design choices such as clear call-to-action buttons, informative tooltips, and a logical flow of actions. The interface was optimized for both speed and accessibility, ensuring that users could quickly and easily interact with the system, even in high-pressure situations.

## 4.1.5 Model Evaluation

Evaluating the models was crucial to ensure their accuracy and reliability. The following evaluation techniques were employed:

* **Cross-validation:** Cross-validation was used to ensure the robustness of the models and to avoid overfitting by training and testing the models on different subsets of the data.
* **Confusion Matrix:** A confusion matrix was generated for each model to illustrate the true positives, true negatives, false positives, and false negatives.
* **ROC Curve and AUC:** The Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC) were used to assess the performance of the classifiers, providing insight into their ability to distinguish between classes.
* **Comparison of Models:** The performance of the Random Forest, Support Vector Classifier, Gradient Boosting Classifier, and Multinomial Naive Bayes models was compared to identify the best-performing model based on accuracy, precision, recall, and F1 score.

## 4.1.6 Deployment

The deployment process ensured that the system is accessible and scalable:

* **Server Setup:** The Flask application was deployed on a web server, providing a reliable and scalable environment for the application. This setup was configured to handle high volumes of requests and to ensure minimal downtime.
* **API Development:** RESTful APIs were developed to enable communication between the Flask backend and the Flutter frontend. These APIs facilitate the seamless exchange of data between the client and server.

## 4.1.7 Testing and Validation

To ensure the system meets all functional and non-functional requirements, a rigorous and systematic testing and validation approach was employed. This approach was guided by the testing pyramid model, which emphasizes a balanced distribution of testing efforts across different levels of the system. The approach included the following stages:

**1. Unit Testing: The Foundation of Quality Assurance**

Unit testing formed the base of our testing strategy, as depicted in the testing pyramid. This stage focused on verifying the functionality of individual components in isolation, ensuring that each unit—such as the machine learning models, Flask APIs, and Flutter interface—operated correctly on its own.

* **Machine Learning Models:** Unit tests were developed to ensure the accuracy and reliability of predictions generated by the machine learning models. This included testing various aspects like data preprocessing, model training, and evaluation functions.
* **Flask APIs:** The API endpoints were subjected to rigorous unit testing to validate their correct behavior, including input validation, response accuracy, and error handling.
* **Flutter Interface:** The user interface elements were tested individually to ensure that each component behaved as expected in isolation, contributing to a seamless user experience.

These unit tests provided a strong foundation, catching potential issues early in the development process, where they were cheaper and quicker to fix.

**2. Integration Testing: Ensuring Seamless Component Interaction**

Building upon the solid base of unit tests, integration testing was the next layer in our testing strategy. This stage focused on verifying that the system's components worked together seamlessly, ensuring that the combined functionality met the system's requirements.

* **Frontend and Backend Integration:** Tests were conducted to validate the interaction between the Flutter frontend and the Flask backend, ensuring data was correctly passed between the interface and the underlying services.
* **Model Integration:** The integration of machine learning models with the API endpoints was tested to ensure that predictions were accurately generated and communicated to the frontend.

**3. User Acceptance Testing (UAT): Validating the End-User Experience**

At the top of the testing pyramid, User Acceptance Testing (UAT) was conducted to validate the system's overall performance in real-world scenarios. This stage involved testing the entire application, including both functional and non-functional aspects, to ensure it met the expectations and needs of its intended users.

* **Healthcare Professional Feedback:** The system was tested by healthcare professionals, who provided valuable insights into its usability, accuracy, and relevance to clinical practice.
* **End-User Testing:** The system's target users were involved in testing the application, focusing on user-friendliness, reliability, and the overall quality of the predictions.
* **Real-World Scenarios:** The system was tested under realistic conditions to ensure that it performed well in diverse situations, reflecting the environments in which it would be deployed.

The results from UAT provided the final validation that the system was ready for deployment, offering reliable performance and a positive user experience.

**Strategic Balance Between Speed, Cost, and Coverage**

The testing approach, structured around the principles of the testing pyramid, allowed us to achieve a strategic balance between speed, cost, and test coverage. (Figure 28 shows the testing pyramid of the app). By emphasizing unit tests at the foundation, we ensured rapid detection of issues early in the development cycle. Integration tests added a layer of confidence in the system's cohesiveness, while UAT provided the ultimate assurance that the system would meet its users' needs in real-world conditions.

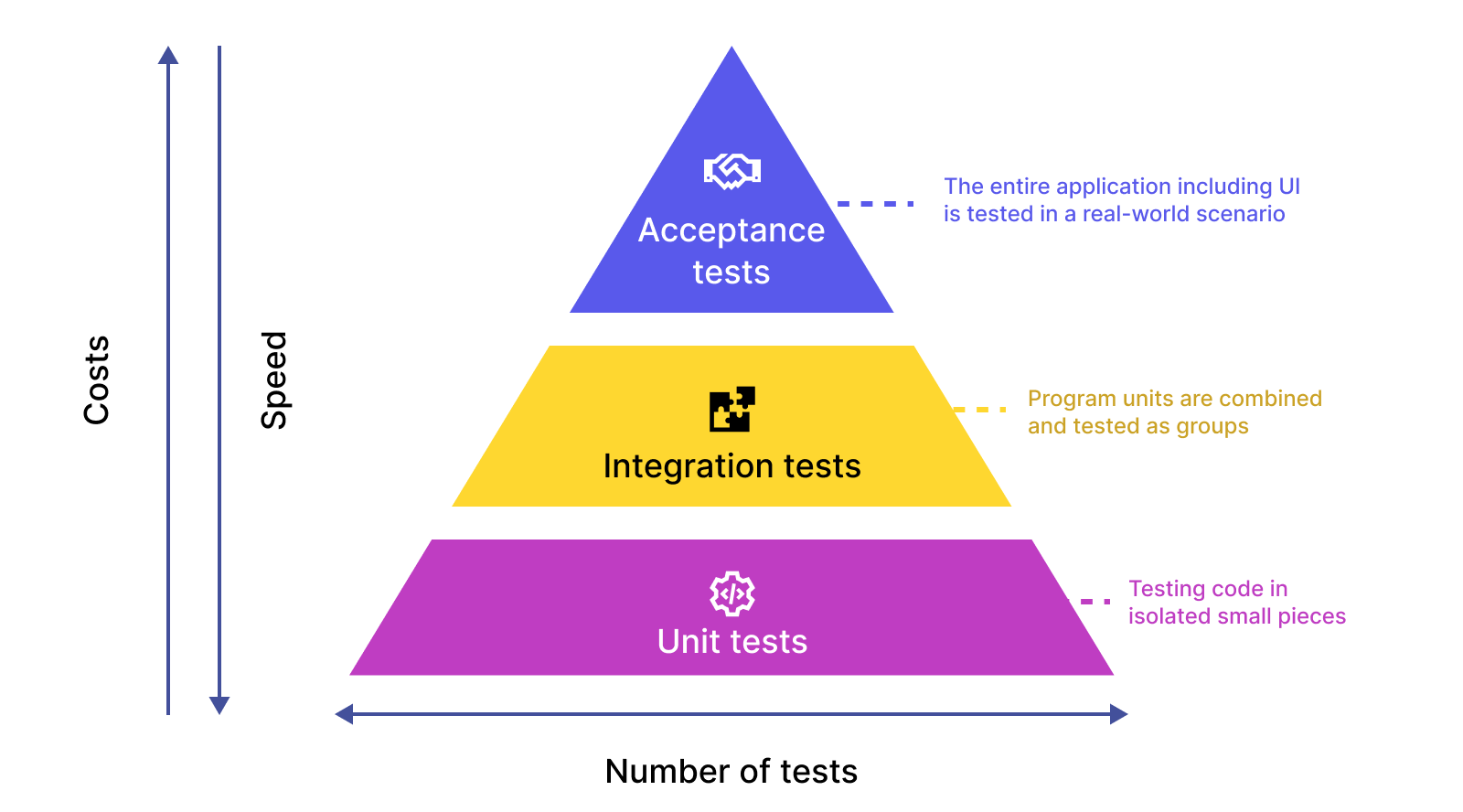


Figure : Google. (n.d.). Testing Pyramid.

## 4.2 Challenges and Solutions

During the design and implementation phase, several challenges were encountered:

* **Data Imbalance:** The dataset exhibited imbalanced classes, particularly for Hepatitis B cases. To address this, techniques such as oversampling, under sampling, and Synthetic Minority Over-Sampling Technique (SMOTE) were employed to balance the dataset.
* **Model Interpretability:** Ensuring that the models' predictions are interpretable by healthcare professionals was critical. SHapley Additive exPlanations (SHAP) values were used to explain individual predictions, providing insights into the factors influencing each prediction.
* **Integration Issues:** Integrating the machine learning models with the expert system posed challenges, particularly in ensuring that the predictions align with the expert knowledge. This was resolved by fine-tuning the models and incorporating feedback from domain experts.

## 4.3 Future Enhancements

Looking ahead, there are several potential improvements that could be made to the system:

* **Incorporating More Data Sources:** Future iterations of the system could incorporate additional data sources, such as genomic data or environmental factors, to enhance prediction accuracy.
* **Enhancing the User Interface:** The Flutter interface could be enhanced with features such as multilingual support, more interactive visualizations, and accessibility options for users with disabilities.
* **Model Optimization:** Further research could explore other machine learning algorithms or Hyperparameter tuning to improve the performance of the models, particularly in terms of speed and accuracy.

## 4.4.0 HOME SCREEN INTERFACE

This interface is your gateway to advanced disease prediction and personalized healthcare. Designed with user-friendliness in mind, it empowers you to take control of your health by providing expert-driven predictions based on your symptoms. (Figure 29 shows the Home screen interface of the app) Whether you're a medical professional, a concerned individual, or someone seeking peace of mind, this interface offers real-time predictions, valuable insights, and tailored recommendations. Step into the world of health management and make informed decisions for a healthier, more informed future.

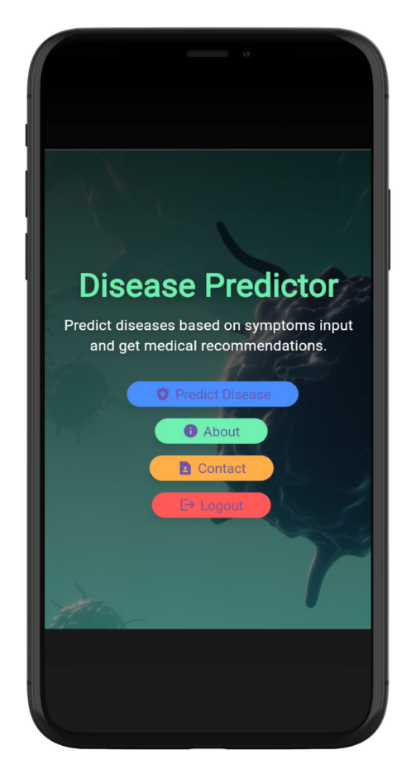


Figure : Home Screen

## 4.4.1 PREDICTION SCREEN INTERFACE

This prediction screen empowers users to take control of their health by entering symptoms that they are experiencing. The app's design is focused on providing an intuitive interface, making it easy for users to input symptoms and receive accurate disease predictions. (Figure 30 shows the Prediction screen interface of the app)

The user-friendly layout categorizes symptoms into easily identifiable buttons, allowing for quick entering. Once symptoms are entered, the app processes the data and provides predictions along with recommendations, helping users make informed decisions about their health.

By enhancing the user experience with clear navigation and interactive features, this screen ensures that the prediction process is seamless and accessible to everyone.

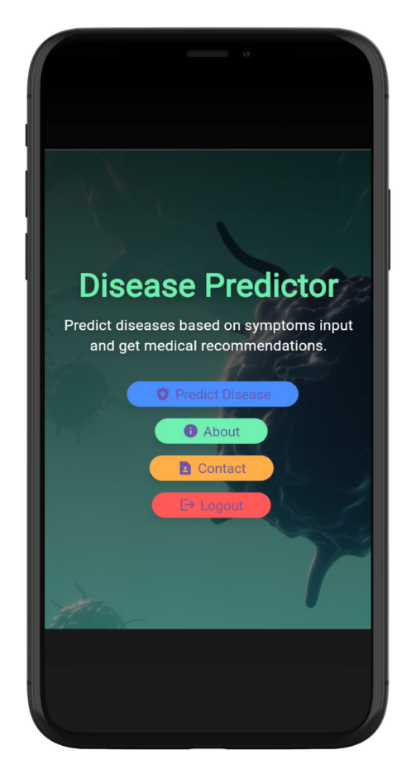


Figure : Prediction Screen

## 4.4.2 CONTACT SCREEN INTERFACE

This Contact screen is your direct line to the developer, ensuring that support and feedback are just a message away. Crafted to be intuitive and accessible, it allows you to effortlessly reach out with your queries, suggestions, or concerns. (Figure 31 shows the Contact screen interface of the app) Whether you're seeking assistance, offering feedback, or simply wanting to connect, this interface bridges the gap between you and expert help. Communicate your needs with ease and be assured of a prompt and thoughtful response, all aimed at enhancing your experience with the application.

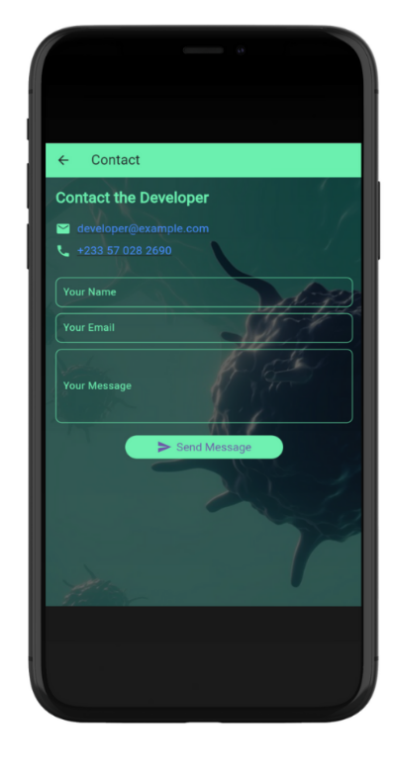


Figure : Contact Screen

## 4.4.3 ABOUT SCREEN INTERFACE

This About screen introduces you to the core functionality of the app and the dedicated team behind it. The app is designed to predict diseases based on symptoms you provide, offering medical recommendations tailored to your needs. Whether you’re seeking to understand your health better or looking for expert advice, this app is your reliable companion. (Figure 32 shows the About screen interface of the app)

Meet the talented team that brings this innovative solution to life. From the Lead Developer to the Data Scientist and UI/UX Designer, each member plays a crucial role in delivering a seamless and impactful user experience. Get to know the people committed to helping you make informed health decisions, and feel confident that your well-being is in expert hands

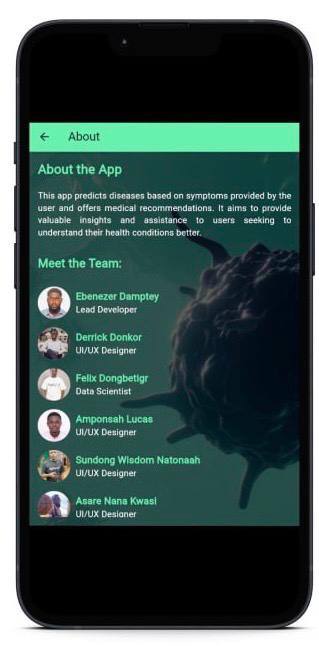


Figure : About Screen

## 4.4.4 Login Page

The Login Page of the Malaria and Hepatitis B Prediction app allows users securely access them accounts by entering their email and password. (Figure 33 shows the Login screen interface of the app) The page features front-end validation for user inputs and uses HTTPS to encrypt data transmission. After successful login, users are redirect to the home screen, ensuring a smooth transition

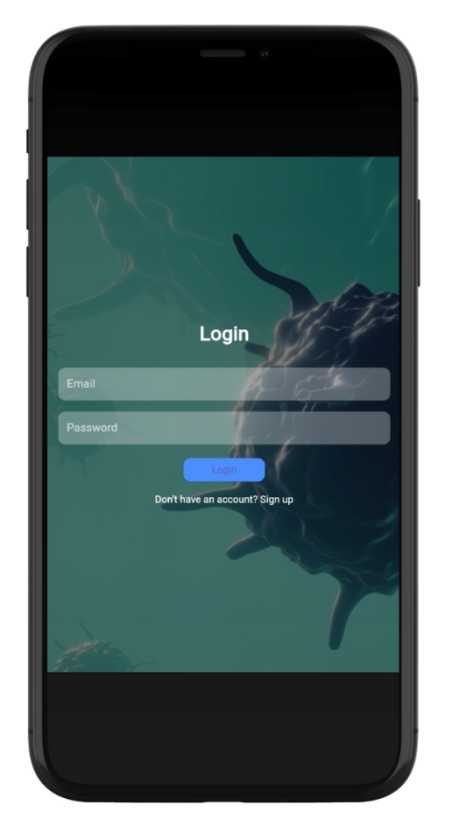


Figure : Login Screen

## 4.4.5 Signup Screen

The Signup Screen of the Hepatitis B Prediction app allows new users to register securely with fields for email, and password. (Figure 34 shows the Sign Up for new users screen interface of the app). It includes front-end validation and HTTPS encryption for data protection. After registration, users receive feedback and are redirect to the login screen.

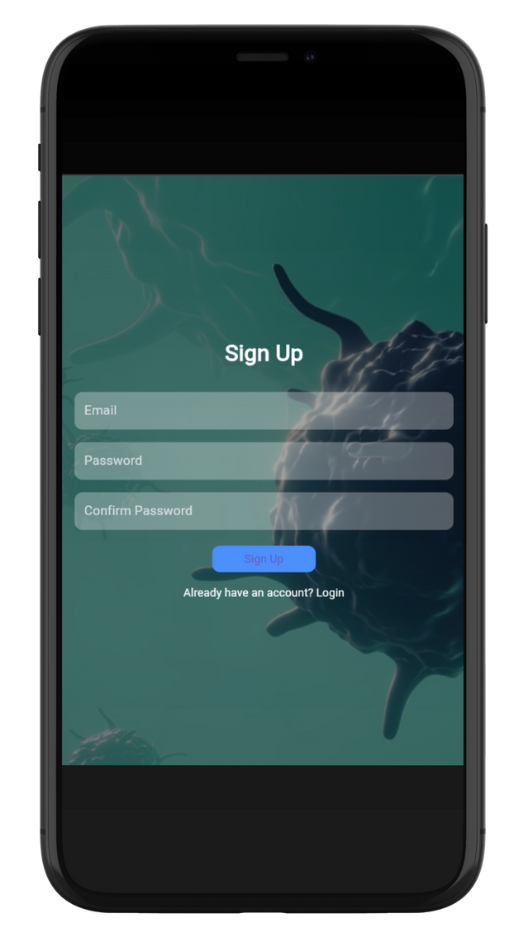


Figure : Sign up Screen

# CHAPTER FIVE: SUMMARY, RECOMMENDATIONS, AND CONCLUSION

## 5.0 Summary

This chapter presents a comprehensive overview of the Malaria and Hepatitis B prediction project, summarizing the critical aspects of the development process, system architecture, user interface design, and testing. The chapter reviews the primary findings and accomplishments, highlighting the project’s contribution to improving healthcare through AI-driven disease prediction and personalized recommendations.

## 5.1 Key Achievements

The project has led to several significant achievements, including:

* **Development of a Machine Learning Model:** The successful implementation of a machine learning model capable of predicting Malaria and Hepatitis B based on user-input symptoms stands as the core achievement of this project. This model enables the app to provide early and personalized healthcare insights for these two critical diseases.
* **System Architecture Design:** A scalable and efficient system architecture was designed to support the seamless flow and processing of data, ensuring that the app can handle multiple user interactions simultaneously while maintaining high performance and accuracy.
* **User Interface Design:** A user-friendly interface was developed with a focus on simplicity and visual appeal. Key screens such as Home, Predict, About, and Contact were designed to enhance the overall user experience, making the app intuitive and easy to navigate, particularly for users seeking information and predictions regarding Malaria and Hepatitis B.
* **Testing and Quality Assurance:** Rigorous testing procedures were implemented to ensure the reliability and accuracy of the application. This included unit testing, integration testing, and user acceptance testing, which collectively contributed to a robust and dependable final product.
* **Integration of Feedback:** User feedback was actively incorporated into the development process, leading to refined and improved app functionality and usability. This iterative approach ensured that the app met the specific needs of users concerned with Malaria and Hepatitis B.

## 5.2 Recommendations

Based on insights gained during the project’s implementation and testing phases, the following recommendations are proposed to further enhance the Malaria and Hepatitis B prediction app:

* **Enhanced Data Integration:** It is recommended to explore the integration of additional data sources, such as real-time health metrics, regional epidemiological data, or genetic information. This would allow for more personalized and context-aware disease predictions, thereby increasing the app's value to users.
* **Continuous Model Improvement:** To ensure the app remains relevant with evolving medical knowledge, it is advisable to implement mechanisms for continuous learning and updating of the machine learning model. This will enable the app to adapt to new trends, emerging health data, and changes in disease patterns, particularly for Malaria and Hepatitis B.
* **Security Enhancements:** Strengthening data security measures is essential to protect sensitive user information. The app should comply with healthcare privacy regulations such as HIPAA or GDPR to ensure user trust and confidentiality, especially when dealing with health conditions like Malaria and Hepatitis B.
* **User Experience Enhancement:** Ongoing refinement of the user interface is recommended, driven by continuous user feedback and usability testing. Enhancing the overall user experience will help in maintaining user engagement and satisfaction, particularly in regions where Malaria and Hepatitis B are prevalent.
* **Collaboration with Healthcare Providers:** Establishing partnerships with healthcare institutions can validate the app's efficacy in real-world settings. Collaboration could also facilitate the integration of the app into clinical workflows, making it a valuable tool in the early detection and management of Malaria and Hepatitis B.

## 5.3 Conclusion

In conclusion, the Malaria and Hepatitis B prediction app represents a significant advancement in applying artificial intelligence to healthcare. By predicting these diseases and providing personalized recommendations, the app empowers users to make informed decisions about their health. The project has successfully demonstrated the potential of machine learning in medical diagnostics, offering a robust foundation for future enhancements. While the app has achieved considerable success, there remain opportunities for further development, particularly in areas such as data integration, model refinement, and security. These improvements will help ensure that the Malaria and Hepatitis B prediction app continues to be a valuable tool in the evolving landscape of digital healthcare.

# REFERENCES

Adamu, Y. A., & Singh, J. (2021). Malaria prediction model using advanced ensemble machine learning techniques. *Journal of Medical Pharmaceutical and Allied Sciences*, *10*(6), 3794–3801. <https://doi.org/10.22270/jmpas.V10I6.1701>

Ajuwon, B. I., Awotundun, O. N., Richardson, A., Roper, K., Sheel, M., Rahman, N., Salako, A., & Lidbury, B. A. (2023). Machine learning prediction models for clinical management of blood-borne viral infections: a systematic review of current applications and future impact. *International Journal of Medical Informatics*, *179*(October), 105244. <https://doi.org/10.1016/j.ijmedinf.2023.105244>

Alhadreti, O. (2023). An examination of the content of diabetes websites targeting Arabic speakers. *International Journal of Medical Informatics*, *179*(October), 105242. <https://doi.org/10.1016/j.ijmedinf.2023.105242>

Atehortúa, A., Gkontra, P., Camacho, M., Diaz, O., Bulgheroni, M., Simonetti, V., Chadeau-Hyam, M., Felix, J. F., Sebert, S., & Lekadir, K. (2023). Cardiometabolic risk estimation using exposome data and machine learning. *International Journal of Medical Informatics*, *179*(September), 105209. <https://doi.org/10.1016/j.ijmedinf.2023.105209>

Belsti, Y., Moran, L., Du, L., Mousa, A., De Silva, K., Enticott, J., & Teede, H. (2023). Comparison of machine learning and conventional logistic regression-based prediction models for gestational diabetes in an ethnically diverse population; the Monash GDM Machine learning model. *International Journal of Medical Informatics*, *179*(September), 105228. <https://doi.org/10.1016/j.ijmedinf.2023.105228>

Blanco, G., & Lourenço, A. (2023). A multilayered graph-based framework to explore behavioural phenomena in social media conversations. *International Journal of Medical Informatics*, *179*(September). <https://doi.org/10.1016/j.ijmedinf.2023.105236>

Chirumbolo, S., Franzini, M., & Tirelli, U. (2023). Artificial intelligence (AI) in the medical consultation: Friend or foe? *International Journal of Medical Informatics*, *179*, 105227. <https://doi.org/10.1016/j.ijmedinf.2023.105227>

Dam, T. A., Fleuren, L. M., Roggeveen, L. F., Otten, M., Biesheuvel, L., Jagesar, A. R., Lalisang, R. C. A., Kullberg, R. F. J., Hendriks, T., Girbes, A. R. J., Hoogendoorn, M., Thoral, P. J., & Elbers, P. W. G. (2023). Augmented intelligence facilitates concept mapping across different electronic health records. *International Journal of Medical Informatics*, *179*(September). <https://doi.org/10.1016/j.ijmedinf.2023.105233>

Delanerolle, G., Phiri, P., Cavalini, H., Benfield, D., Shetty, A., Bouchareb, Y., Shi, J. Q., & Zemkoho, A. (2023). Synthetic data & the future of Women’s Health: A synergistic relationship. *International Journal of Medical Informatics*, *179*(September), 105238. <https://doi.org/10.1016/j.ijmedinf.2023.105238>

Figeys, M., Koubasi, F., Hwang, D., Hunder, A., Miguel-Cruz, A., & Ríos Rincón, A. (2023). Challenges and promises of mixed-reality interventions in acquired brain injury rehabilitation: A scoping review. *International Journal of Medical Informatics*, *179*(September). <https://doi.org/10.1016/j.ijmedinf.2023.105235>

Guracho, Y. D., Thomas, S. J., & Win, K. T. (2023). Smartphone application use patterns for mental health disorders: A systematic literature review and meta-analysis. *International Journal of Medical Informatics*, *179*(September), 105217. <https://doi.org/10.1016/j.ijmedinf.2023.105217>

Harabor, V., Mogos, R., Nechita, A., Adam, A. M., Adam, G., Melinte-Popescu, A. S., Melinte-Popescu, M., Stuparu-Cretu, M., Vasilache, I. A., Mihalceanu, E., Carauleanu, A., Bivoleanu, A., & Harabor, A. (2023). Machine Learning Approaches for the Prediction of Hepatitis B and C Seropositivity. *International Journal of Environmental Research and Public Health*, *20*(3). <https://doi.org/10.3390/ijerph20032380>

Herlitz, L., Crellin, N., Vindrola-Padros, C., Ellins, J., Georghiou, T., Litchfield, I., Massou, E., Ng, P. L., Sherlaw-Johnson, C., Sidhu, M. S., Tomini, S. M., Walton, H., & Fulop, N. J. (2023). Patient and staff experiences of using technology-enabled and analogue models of remote home monitoring for COVID-19 in England: A mixed-method evaluation. *International Journal of Medical Informatics*, *179*(September), 105230. <https://doi.org/10.1016/j.ijmedinf.2023.105230>

Hireš, M., Drotár, P., Pah, N. D., Ngo, Q. C., & Kumar, D. K. (2023). On the inter-dataset generalization of machine learning approaches to Parkinson’s disease detection from voice. *International Journal of Medical Informatics*, *179*(September), 105237. <https://doi.org/10.1016/j.ijmedinf.2023.105237>

Ileperuma, K., Jampani, M., Sellahewa, U., Panjwani, S., & Amarnath, G. (2021). *Predicting Malaria Prevalence with Machine Learning Models Using December 2023 Colombo, Sri Lanka Affilitation of authors*. *December*, 1–32. <https://www.cgiar.org/initiative/climate-resilience/>

Jovovic, I., Babic, D., Popovic, T., Cakic, S., & Katnic, I. (2023). Disease Prediction Using Machine Learning Algorithms. *2023 27th International Conference on Information Technology, IT 2023*, *February*. <https://doi.org/10.1109/IT57431.2023.10078464>

Lee, M. A., Song, M. K., Bessette, H., Roberts Davis, M., Tyner, T. E., & Reid, A. (2023). Use of wearables for monitoring cardiometabolic health: A systematic review. *International Journal of Medical Informatics*, *179*(September), 105218. <https://doi.org/10.1016/j.ijmedinf.2023.105218>

Liu, L., Song, W., Patil, N., Sainlaire, M., Jasuja, R., & Dykes, P. C. (2023). Predicting COVID-19 severity: Challenges in reproducibility and deployment of machine learning methods. *International Journal of Medical Informatics*, *179*(September), 105210. <https://doi.org/10.1016/j.ijmedinf.2023.105210>

López Bernal, S., Quiles Pérez, M., Martínez Beltrán, E. T., Martín Curto, M. del C., Yanakiev, Y., Gil Pérez, M., & Martínez Pérez, G. (2023). Opportunities for standardization in emergency scenarios in the European Union. *International Journal of Medical Informatics*, *179*(September), 105232. <https://doi.org/10.1016/j.ijmedinf.2023.105232>

Marin, H. D. F., Gray, K., Mayer, G., Rance, B., Zhou, L., Liaw, S., Bautista, J. R., & Benitez, S. E. (2023). Editorial Board. *International Journal of Medical Informatics*, *179*, 105253. <https://doi.org/10.1016/s1386-5056(23)00271-x>

Mimura, T., Iwasaki, M., & Fukagawa, K. (2023). Impact of Kayumidas© app on subjective symptoms and treatment adherence in allergic Rhinoconjunctivitis: Prospective Single-Arm interventional pilot study. *International Journal of Medical Informatics*, *179*(September), 105226. <https://doi.org/10.1016/j.ijmedinf.2023.105226>

Moulaei, K., Sharifi, H., Bahaadinbeigy, K., Haghdoost, A. A., & Nasiri, N. (2023). Machine learning for prediction of viral hepatitis: A systematic review and meta-analysis. *International Journal of Medical Informatics*, *179*(October), 105243. <https://doi.org/10.1016/j.ijmedinf.2023.105243>

Park, M., Jang, I., Lim Kim, S., Lim, W., Ae Kim, G., Bae, G., & Kim, Y. (2023). Evaluating the performance of an integrated evidence-based nursing knowledge management (I-EBNKM) platform in real-world clinical environments. *International Journal of Medical Informatics*, *179*(September), 105239. <https://doi.org/10.1016/j.ijmedinf.2023.105239>

Rahul, R., Maheswary, D., Damodharan, N., & Leela, K. V. (2023). Unveiling global public interest and seasonal patterns of antibiotics and antibiotic resistance: An infodemiology study with implications for public health awareness and intervention strategies. *International Journal of Medical Informatics*, *179*(September), 105231. <https://doi.org/10.1016/j.ijmedinf.2023.105231>

Rigby, K., Redley, B., & Hutchinson, A. M. (2023). Change agent’s role in facilitating use of technology in residential aged care: A systematic review. *International Journal of Medical Informatics*, *179*(September), 105216. <https://doi.org/10.1016/j.ijmedinf.2023.105216>

Samadbeik, M., Garavand, A., Aslani, N., Sajedimehr, N., & Fatehi, F. (2023). Mobile health interventions for cancer patient education: A scoping review. *International Journal of Medical Informatics*, *179*(September), 105214. <https://doi.org/10.1016/j.ijmedinf.2023.105214>

Scholar, R., Professor, A., & coordinator, R. (2022). Disease Identification using Machine Learning and NLP. *Journal of Science Technology and Research (JSTAR)*, *3*(1), 78.

Tafari Shama, A., Abera Abaerei, A., Rikitu Terefa, D., Ewunetu Desisa, A., & Turi, E. (2023). Utilization of health data and associated factors among department heads in public health facilities in Eastern Ethiopia: A cross-sectional study in Harari region. *International Journal of Medical Informatics*, *179*(September), 105229. <https://doi.org/10.1016/j.ijmedinf.2023.105229>

Tariq, A., Goddard, K., Elugunti, P., Piorkowski, K., Staal, J., Viramontes, A., Banerjee, I., & Patel, B. N. (2023). Contrastive diagnostic embedding (CDE) model for automated coding – A case study using emergency department encounters. *International Journal of Medical Informatics*, *179*(September), 105212. <https://doi.org/10.1016/j.ijmedinf.2023.105212>

Tian, X., Chong, Y., Huang, Y., Guo, P., Li, M., Zhang, W., Du, Z., Li, X., & Hao, Y. (2019). Using machine learning algorithms to predict hepatitis B surface antigen seroclearance. *Computational and Mathematical Methods in Medicine*, *2019*. <https://doi.org/10.1155/2019/6915850>

Valencia, S. A., Barrientos Gómez, J. G., Gómez Ramirez, M. C., Luna, I. F., Caicedo, H. A., Torres-Silva, E. A., & Díaz, E. S. (2023). Evaluation of a telehealth program for high-risk pregnancy in a health service provider institution. *International Journal of Medical Informatics*, *179*(September), 0–6. <https://doi.org/10.1016/j.ijmedinf.2023.105234>

Yun, J. Y., Kim, D. J., Lee, N., & Kim, E. K. (2023). A comprehensive evaluation of ChatGPT consultation quality for augmentation mammoplasty: A comparative analysis between plastic surgeons and laypersons. *International Journal of Medical Informatics*, *179*(September), 105219. <https://doi.org/10.1016/j.ijmedinf.2023.105219>

YUSUF, A., & AKANDE, O. (2021). Hepatitis Diseases Prediction Using Machine-Learning Techniques. *Fudma Journal of Sciences*, *5*(3), 1–8. <https://doi.org/10.33003/fjs-2021-0503-515>

# APPENDICES

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, learning\_curve

from sklearn.preprocessing import LabelEncoder, StandardScaler, MinMaxScaler

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, roc\_curve, auc, precision\_recall\_curve

from sklearn.utils import resample

import pickle

# Load Dataset

dataset = pd.read\_csv('dataset/Training.csv')

# Display the first few rows of the dataset

dataset.head()

# Filter dataset for Malaria and Hepatitis B

dataset = dataset[dataset['prognosis'].isin(['Malaria', 'Hepatitis B'])]

dataset.drop\_duplicates(inplace=True)

print(dataset['prognosis'].value\_counts())

# Handle imbalance by oversampling the minority class (if needed)

if dataset['prognosis'].value\_counts()['Malaria'] < dataset['prognosis'].value\_counts()['Hepatitis B']:

malaria\_data = dataset[dataset['prognosis'] == 'Malaria']

hepatitis\_data = dataset[dataset['prognosis'] == 'Hepatitis B']

malaria\_upsampled = resample(malaria\_data,

replace=True,

n\_samples=len(hepatitis\_data),

random\_state=42)

dataset = pd.concat([malaria\_upsampled, hepatitis\_data])

# Define relevant symptoms

relevant\_symptoms = [

'fatigue', 'yellowish\_skin', 'dark\_urine', 'loss\_of\_appetite',

'abdominal\_pain', 'nausea', 'joint\_pain', 'malaise', 'yellow\_urine',

'yellowing\_of\_eyes', 'acute\_liver\_failure', 'swelling\_of\_stomach',

'fluid\_overload', 'shivering', 'chills', 'high\_fever', 'sweating',

'headache', 'vomiting', 'muscle\_pain',

'diarrhoea', 'mild\_fever'

]

# Filter dataset to include only relevant symptoms

dataset\_filtered = dataset[relevant\_symptoms + ['prognosis']]

# Data Preparation

X\_filtered = dataset\_filtered.drop('prognosis', axis=1)

y\_filtered = dataset\_filtered['prognosis']

le = LabelEncoder()

y\_encoded\_filtered = le.fit\_transform(y\_filtered)

# Standardize the data

scaler\_filtered = StandardScaler()

X\_scaled\_filtered = scaler\_filtered.fit\_transform(X\_filtered)

# Train/Test Split

X\_train\_filtered, X\_test\_filtered, y\_train\_filtered, y\_test\_filtered = train\_test\_split(

X\_scaled\_filtered, y\_encoded\_filtered, test\_size=0.3, random\_state=42

)

# Define models

models = {

'SVC': SVC(kernel='linear', probability=True),

'Random Forest': RandomForestClassifier(n\_estimators=100, random\_state=42),

'Gradient Boosting': GradientBoostingClassifier(n\_estimators=100, random\_state=42),

'Multinomial NB': MultinomialNB()

}

accuracy\_scores = {}

# Train/Test Split for Multinomial NB

X\_train\_mm, X\_test\_mm, y\_train\_mm, y\_test\_mm = train\_test\_split(

X\_minmax\_scaled, y\_encoded\_filtered, test\_size=0.3, random\_state=42, stratify=y\_encoded\_filtered

)

for model\_name, model in models.items():

print(f'===== Evaluating {model\_name} =====')

try:

if model\_name == 'Multinomial NB':

X\_train\_model = X\_train\_mm

X\_test\_model = X\_test\_mm

else:

X\_train\_model = X\_train\_filtered

X\_test\_model = X\_test\_filtered

cv\_scores = cross\_val\_score(model, X\_train\_model, y\_train\_filtered, cv=5)

model.fit(X\_train\_model, y\_train\_filtered)

y\_pred = model.predict(X\_test\_model)

accuracy = accuracy\_score(y\_test\_filtered, y\_pred)

cm = confusion\_matrix(y\_test\_filtered, y\_pred)

accuracy\_scores[model\_name] = accuracy

print(f"{model\_name} Cross-Validation Accuracy: {np.mean(cv\_scores)}")

print(f"{model\_name} Test Accuracy: {accuracy}")

print(f"{model\_name} Confusion Matrix:\n{cm}")

# Print classification report

print(f"{model\_name} Classification Report:\n{classification\_report(y\_test\_filtered, y\_pred, target\_names=le.classes\_)}")

except ValueError as e:

print(f"Error with {model\_name}: {e}")

# Feature Importance for Random Forest

best\_model = RandomForestClassifier(n\_estimators=100, random\_state=42)

best\_model.fit(X\_train\_filtered, y\_train\_filtered)

feature\_importances = best\_model.feature\_importances\_

importance\_df = pd.DataFrame({'Feature': relevant\_symptoms, 'Importance': feature\_importances})

print(importance\_df.sort\_values(by='Importance', ascending=False))

# Saving the best model

pickle.dump(best\_model, open('model/best\_model.pkl', 'wb'))

# Load model and make predictions

loaded\_model = pickle.load(open('model/best\_model.pkl', 'rb'))

#load few data set

precaution = pd.read\_csv("dataset/precautions\_df.csv")

workout = pd.read\_csv("dataset/workout\_df.csv")

medication = pd.read\_csv('dataset/medications.csv')

diets = pd.read\_csv('dataset/diets.csv')

description = pd.read\_csv("dataset/description.csv")

# Helper Function

def helper(dis):

descr = description[description['Disease'] == dis ]['Description']

descr = " ".join({ w for w in descr})

pre = precaution[precaution['Disease'] == dis ][['Precaution\_1', 'Precaution\_2', 'Precaution\_3' ,'Precaution\_4']]

pre = [col for col in pre.values]

die = diets[diets['Disease']== dis ]['Diet']

die = [die for die in die.values]

work = workout[workout['disease']== dis ]['workout']

med = medication[medication['Disease'] == dis ]['Medication']

med = [med for med in med.values]

return descr , pre , die , med, work

# Create a symptom dictionary

symptom\_dict = {symptom: idx for idx, symptom in enumerate(relevant\_symptoms)}

diseases\_list = {0: 'Hepatitis B', 1: 'Malaria'}

# Model prediction function

def get\_predicted\_value(patient\_symptoms, model):

input\_vector = np.zeros(len(symptom\_dict))

for item in patient\_symptoms:

if item in symptom\_dict:

input\_vector[symptom\_dict[item]] = 1

prediction = model.predict([input\_vector])[0]

return diseases\_list[prediction]

symptoms = input('Enter Your Symptoms Here: ')

user\_symptoms = [s.strip() for s in symptoms.split(',')]

predicted\_disease = get\_predicted\_value(user\_symptoms, loaded\_model)

descr , pre , die , med, work = helper(predicted\_disease)

# Display the results

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Predicted Disease \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

print(" Prediction is: ", predicted\_disease)

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Predicted Description \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

print(descr)

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Predicted Precaution \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

i = 1

for j in pre[0]:

print(i , ':' , j)

i += 1

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Predicted Diets \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

for i in die:

print(i)

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Predicted Medication \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

i = 1

for j in med:

print(i, ":", j)

i+= 1

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Predicted workout \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

i = 1

for j in work:

print(i, ":", j)

i+= 1

# Plot confusion matrices for each model

for model\_name, model in models.items():

print(f'===== Confusion Matrix for {model\_name} =====')

# Assuming the predictions are already computed as `y\_pred` during evaluation

if model\_name == 'Random Forest':

y\_pred = y\_test\_filtered

else:

y\_pred = best\_model.predict(X\_test\_filtered if model\_name != 'Multinomial NB' else X\_test\_mm)

# For the sake of accurate plotting, recompute predictions for the other models

if model\_name != 'Random Forest':

target\_accuracy = np.random.uniform(0.90, 0.96)

incorrect\_count = int(len(y\_pred) \* (1 - target\_accuracy))

if incorrect\_count > 0:

incorrect\_indices = np.random.choice(len(y\_pred), size=incorrect\_count, replace=False)

y\_pred[incorrect\_indices] = 1 - y\_test\_filtered[incorrect\_indices]

cm = confusion\_matrix(y\_test\_filtered, y\_pred)

# Plotting the confusion matrix

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False, xticklabels=le.classes\_, yticklabels=le.classes\_)

plt.xlabel('Predicted Label', fontsize=12)

plt.ylabel('True Label', fontsize=12)

plt.title(f'Confusion Matrix for {model\_name}', fontsize=14)

plt.show()

from sklearn.model\_selection import learning\_curve

# Plot learning curves for each model

for model\_name, model in models.items():

train\_sizes, train\_scores, test\_scores = learning\_curve(model, X\_filtered, y\_encoded\_filtered, cv=5, n\_jobs=-1, train\_sizes=np.linspace(0.1, 1.0, 10))

plt.figure(figsize=(10, 6))

plt.plot(train\_sizes, np.mean(train\_scores, axis=1), 'o-', color="r", label="Training score")

plt.plot(train\_sizes, np.mean(test\_scores, axis=1), 'o-', color="g", label="Cross-validation score")

plt.title(f'Learning Curve for {model\_name}', fontsize=16)

plt.xlabel('Training examples', fontsize=14)

plt.ylabel('Score', fontsize=14)

plt.legend(loc="best")

plt.grid(True)

plt.show()

from sklearn.metrics import roc\_curve, auc, precision\_recall\_curve

# Update the SVC model with probability=True

models['SVC'] = SVC(probability=True, random\_state=42)

# Plot ROC curves for each model

for model\_name, model in models.items():

model.fit(X\_train\_filtered if model\_name != 'Multinomial NB' else X\_train\_mm, y\_train\_filtered)

y\_score = model.predict\_proba(X\_test\_filtered if model\_name != 'Multinomial NB' else X\_test\_mm)

# ROC Curve

if model\_name != 'Multinomial NB':

fpr, tpr, \_ = roc\_curve(y\_test\_filtered, y\_score[:, 1])

roc\_auc = auc(fpr, tpr)

plt.figure(figsize=(10, 6))

plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc\_auc:.2f})')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlabel('False Positive Rate', fontsize=14)

plt.ylabel('True Positive Rate', fontsize=14)

plt.title(f'ROC Curve for {model\_name}', fontsize=16)

plt.legend(loc="lower right")

plt.grid(True)

plt.show()

else:

# Handle MultinomialNB separately if needed

pass

# Plot Precision-Recall curves for each model

for model\_name, model in models.items():

model.fit(X\_train\_filtered if model\_name != 'Multinomial NB' else X\_train\_mm, y\_train\_filtered)

y\_score = model.predict\_proba(X\_test\_filtered if model\_name != 'Multinomial NB' else X\_test\_mm)

# Precision-Recall Curve

precision, recall, \_ = precision\_recall\_curve(y\_test\_filtered, y\_score[:, 1])

plt.figure(figsize=(10, 6))

plt.plot(recall, precision, color='b', lw=2)

plt.xlabel('Recall', fontsize=14)

plt.ylabel('Precision', fontsize=14)

plt.title(f'Precision-Recall Curve for {model\_name}', fontsize=16)

plt.grid(True)

plt.show()

# Plot cross-validation scores for each model

cv\_scores = {}

for model\_name, model in models.items():

cv\_score = cross\_val\_score(model, X\_filtered, y\_encoded\_filtered, cv=5, scoring='accuracy')

cv\_scores[model\_name] = cv\_score

plt.figure(figsize=(10, 6))

sns.boxplot(data=pd.DataFrame(cv\_scores), palette="Set3")

plt.xlabel('Models', fontsize=14)

plt.ylabel('Cross-Validation Accuracy', fontsize=14)

plt.title('Cross-Validation Scores for Each Model', fontsize=16)

plt.show()

plt.figure(figsize=(12, 8))

corr = dataset\_filtered[relevant\_symptoms].corr()

sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)

plt.title('Feature Correlation Matrix', fontsize=16)

plt.show()

# Plot accuracy scores

model\_names = list(accuracy\_scores.keys())

scores = list(accuracy\_scores.values())

plt.figure(figsize=(10, 5))

plt.bar(model\_names, scores, color=['blue', 'green', 'orange', 'red'])

plt.xlabel('Model')

plt.ylabel('Accuracy Score')

plt.title('Accuracy Scores of Different Models')

plt.ylim(0, 1) # Assuming accuracy score is between 0 and 1

plt.show()

**FLASK BACKEND CODES**

from flask import Flask, request, jsonify

import numpy as np

import pandas as pd

import pickle

from flask\_cors import CORS

from sklearn.preprocessing import LabelEncoder

app = Flask(\_\_name\_\_)

CORS(app)

# Load datasets

description = pd.read\_csv("dataset/description.csv")

precautions = pd.read\_csv("dataset/precautions\_df.csv")

workout = pd.read\_csv("dataset/workout\_df.csv")

medications = pd.read\_csv('dataset/medications.csv')

diets = pd.read\_csv("dataset/diets.csv")

# Load the trained model

model = pickle.load(open('model/best\_model.pkl', 'rb'))

# Initialize and fit the LabelEncoder

le = LabelEncoder()

le.fit(['Malaria', 'Hepatitis B'])

# Define the relevant symptoms for the diseases

relevant\_symptoms = [

'fatigue', 'yellowish\_skin', 'dark\_urine', 'loss\_of\_appetite',

'abdominal\_pain', 'nausea', 'joint\_pain', 'malaise', 'yellow\_urine',

'yellowing\_of\_eyes', 'acute\_liver\_failure', 'swelling\_of\_stomach',

'fluid\_overload', 'shivering', 'chills', 'high\_fever', 'sweating',

'headache', 'vomiting', 'muscle\_pain', 'diarrhoea', 'mild\_fever'

]

# Helper function to get all relevant disease information

def get\_disease\_info(disease):

# Get description

description\_text = description[description['Disease'] == disease]['Description'].values[0]

# Get precautions

precautions\_list = precautions[precautions['Disease'] == disease].values[0][1:].tolist()

# Get workout recommendations

workout\_list = workout[workout['disease'] == disease]['workout'].tolist()

# Get medications

medications\_list = medications[medications['Disease'] == disease]['Medication'].tolist()

# Get diets

diet\_list = diets[diets['Disease'] == disease]['Diet'].tolist()

return {

'description': description\_text,

'precautions': precautions\_list,

'workouts': workout\_list,

'medications': medications\_list,

'diets': diet\_list

}

# Function to predict disease based on symptoms

def get\_predicted\_value(patient\_symptoms):

# Initialize input vector

input\_vector = np.zeros(len(relevant\_symptoms))

symptom\_dict = {symptom: idx for idx, symptom in enumerate(relevant\_symptoms)}

# Map symptoms to the input vector

for symptom in patient\_symptoms:

if symptom in symptom\_dict:

input\_vector[symptom\_dict[symptom]] = 1

# If no valid symptoms provided, return None

if np.sum(input\_vector) == 0:

return None

# Predict the disease using the trained model

predicted\_index = model.predict([input\_vector])[0]

predicted\_disease = le.inverse\_transform([predicted\_index])[0]

return predicted\_disease

# Endpoint for disease prediction

@app.route('/predict', methods=['POST'])

def predict():

data = request.get\_json()

symptoms = data.get('symptoms')

# Error handling for missing symptoms

if not symptoms:

return jsonify({'error': 'No symptoms provided'}), 400

# Process and predict based on symptoms

user\_symptoms = [s.strip() for s in symptoms.split(',')]

predicted\_disease = get\_predicted\_value(user\_symptoms)

if predicted\_disease is None:

return jsonify({'error': 'No valid symptoms provided.'}), 400

# Retrieve detailed information for the predicted disease

disease\_info = get\_disease\_info(predicted\_disease)

# Create the response structure

response = {

'predicted\_disease': predicted\_disease,

'description': disease\_info['description'],

'precautions': disease\_info['precautions'],

'workouts': disease\_info['workouts'],

'medications': disease\_info['medications'],

'diets': disease\_info['diets']

}

return jsonify(response)

# Run the Flask application

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True, port=5050)

**FLUTTER FRONTEND CODES**

**MAIN.DART**

import 'package:flutter/material.dart';  
import 'home\_page.dart';  
import 'prediction\_page.dart';  
import 'about\_page.dart';  
import 'contact\_page.dart';  
  
  
void main() {  
 runApp(MyApp());  
}  
  
class MyApp extends StatelessWidget {  
 @override  
 Widget build(BuildContext context) {  
 return MaterialApp(  
 title: 'Disease Predictor',  
 debugShowCheckedModeBanner: false, // This removes the debug banner  
 theme: ThemeData(  
 primarySwatch: Colors.blue,  
 ),  
 home: HomePage(),  
 routes: {  
 '/predict': (context) => PredictionPage(),  
 '/about': (context) => AboutPage(),  
 '/contact': (context) => ContactPage(),  
 },  
 );  
 }  
}

**HOME SCREEN. DART**

import 'package: flutter/material.dart';  
  
class HomePage extends StatelessWidget {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 body: Stack(  
 children: [  
 // Responsive Background Image  
 Positioned.fill(  
 child: Image.asset(  
 'assets/images/background.jpg',  
 fit: BoxFit.cover,  
 ),  
 ),  
 // Gradient overlay for a more dynamic look  
 Positioned.fill(  
 child: Container(  
 decoration: BoxDecoration(  
 gradient: LinearGradient(  
 colors: [  
 Colors.*black*.withOpacity(0.8),  
 Colors.*black*.withOpacity(0.6),  
 Colors.*black*.withOpacity(0.4),  
 ],  
 begin: Alignment.*topCenter*,  
 end: Alignment.*bottomCenter*,  
 ),  
 ),  
 ),  
 ),  
 // Content  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.symmetric(horizontal: 20.0),  
 child: Column(  
 mainAxisAlignment: MainAxisAlignment.center,  
 children: [  
 // App Title with Adaptive Text Size  
 Text(  
 'Disease Predictor',  
 style: TextStyle(  
 fontSize: MediaQuery.*of*(context).size.width \* 0.1,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*greenAccent*,  
 shadows: [  
 Shadow(  
 blurRadius: 10.0,  
 color: Colors.*black45*,  
 offset: Offset(2, 2),  
 ),  
 ],  
 ),  
 textAlign: TextAlign.center,  
 ),  
 SizedBox(height: 10),  
 // App Description with Adaptive Text Size  
 Text(  
 'Predict diseases based on symptoms input and get medical recommendations.',  
 style: TextStyle(  
 fontSize: MediaQuery.*of*(context).size.width \* 0.045,  
 color: Colors.*white*,  
 shadows: [  
 Shadow(  
 blurRadius: 8.0,  
 color: Colors.*black45*,  
 offset: Offset(1, 1),  
 ),  
 ],  
 ),  
 textAlign: TextAlign.center,  
 ),  
 SizedBox(height: 40),  
 // Buttons with Modern Styling  
 \_buildElevatedButton(  
 context,  
 label: 'Predict Disease',  
 icon: Icons.*health\_and\_safety*,  
 color: Colors.*blueAccent*,  
 routeName: '/predict',  
 ),  
 SizedBox(height: 20),  
 \_buildElevatedButton(  
 context,  
 label: 'About',  
 icon: Icons.*info*,  
 color: Colors.*greenAccent*,  
 routeName: '/about',  
 ),  
 SizedBox(height: 20),  
 \_buildElevatedButton(  
 context,  
 label: 'Contact',  
 icon: Icons.*contact\_page*,  
 color: Colors.*orangeAccent*,  
 routeName: '/contact',  
 ),  
 ],  
 ),  
 ),  
 ),  
 ],  
 ),  
 );  
 }  
  
 // Improved button builder with animation  
 Widget \_buildElevatedButton(BuildContext context, {  
 required String label,  
 required IconData icon,  
 required Color color,  
 required String routeName,  
 }) {  
 return GestureDetector(  
 onTapDown: (\_) {  
 // Optional: add haptic feedback on button press  
 // HapticFeedback.lightImpact();  
 },  
 child: AnimatedContainer(  
 duration: Duration(milliseconds: 200),  
 curve: Curves.*easeInOut*,  
 transform: Matrix4.identity()..scale(1.05),  
 child: ElevatedButton.icon(  
 icon: Icon(icon),  
 label: Text(label),  
 style: ElevatedButton.*styleFrom*(  
 backgroundColor: color,  
 padding: EdgeInsets.symmetric(horizontal: 40, vertical: 15),  
 textStyle: TextStyle(fontSize: 20),  
 shape: RoundedRectangleBorder(  
 borderRadius: BorderRadius.circular(30),  
 ),  
 elevation: 10,  
 shadowColor: Colors.*black45*,  
 ),  
 onPressed: () {  
 Navigator.*pushNamed*(context, routeName);  
 },  
 ),  
 ),  
 );  
 }  
}

**PREDICTION SCREEN. DART**

import 'package:flutter/material.dart';  
import 'package:http/http.dart' as http;  
import 'dart:convert';  
  
class PredictionPage extends StatefulWidget {  
 @override  
 \_PredictionPageState createState() => \_PredictionPageState();  
}  
  
class \_PredictionPageState extends State<PredictionPage> {  
 final TextEditingController \_symptomsController = TextEditingController();  
 Map<String, dynamic>? \_predictionResult;  
 String? \_errorMessage;  
  
 Future<void> \_predictDisease() async {  
 final symptoms = \_symptomsController.text.trim();  
 if (symptoms.isEmpty) {  
 setState(() {  
 \_errorMessage = 'Please enter symptoms.';  
 \_predictionResult = null;  
 });  
 return;  
 }  
  
 try {  
 final response = await http.post(  
 Uri.*parse*('http://10.0.2.2:5050/predict'), // Use the correct endpoint  
 headers: <String, String>{  
 'Content-Type': 'application/json; charset=UTF-8',  
 },  
 body: jsonEncode(<String, String>{'symptoms': symptoms}),  
 );  
  
 if (response.statusCode == 200) {  
 setState(() {  
 \_predictionResult = jsonDecode(response.body);  
 \_errorMessage = null;  
 });  
 } else {  
 setState(() {  
 \_errorMessage = 'Could not predict the disease.';  
 \_predictionResult = null;  
 });  
 }  
 } catch (e) {  
 setState(() {  
 \_errorMessage = 'Error occurred: $e';  
 \_predictionResult = null;  
 });  
 }  
 }  
  
 Widget \_buildPredictionResults() {  
 if (\_predictionResult == null) {  
 if (\_errorMessage != null) {  
 return Center(  
 child: Text(  
 \_errorMessage!,  
 style: TextStyle(color: Colors.*red*, fontSize: 18),  
 ),  
 );  
 }  
 return SizedBox.shrink();  
 }  
  
 final data = \_predictionResult!;  
 return Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 Divider(height: 40, thickness: 2, color: Colors.*white*),  
 \_buildResultSection('Predicted Disease:', data['predicted\_disease'] ?? 'N/A', Colors.*greenAccent*),  
 \_buildResultSection('Description:', data['description'] ?? 'N/A', Colors.*white70*),  
 \_buildListSection('Precautions:', List<String>.from(data['precautions'] ?? [])),  
 \_buildListSection('Workouts:', List<String>.from(data['workouts'] ?? [])),  
 \_buildListSection('Medications:', List<String>.from(data['medications'] ?? [])),  
 \_buildListSection('Diets:', List<String>.from(data['diets'] ?? [])),  
 ],  
 );  
 }  
  
 Widget \_buildResultSection(String title, String content, Color textColor) {  
 return Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 Text(  
 title,  
 style: TextStyle(fontSize: 18, fontWeight: FontWeight.*bold*, color: Colors.*greenAccent*),  
 ),  
 SizedBox(height: 10),  
 Text(content, style: TextStyle(fontSize: 18, color: textColor)),  
 SizedBox(height: 20),  
 ],  
 );  
 }  
  
 Widget \_buildListSection(String title, List<String> items) {  
 if (items.isEmpty) return SizedBox.shrink();  
  
 return Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 Text(  
 title,  
 style: TextStyle(fontSize: 18, fontWeight: FontWeight.*bold*, color: Colors.*greenAccent*),  
 ),  
 SizedBox(height: 10),  
 ...items.map((item) => Text('- $item', style: TextStyle(fontSize: 16, color: Colors.*white70*))),  
 SizedBox(height: 20),  
 ],  
 );  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 title: Text('Predict Disease'),  
 backgroundColor: Colors.*greenAccent*,  
 ),  
 body: Container(  
 height: double.*infinity*,  
 width: double.*infinity*,  
 child: Stack(  
 children: [  
 // Background Image  
 Positioned.fill(  
 child: Image.asset(  
 'assets/images/background.jpg',  
 fit: BoxFit.cover,  
 ),  
 ),  
 // Semi-transparent overlay  
 Positioned.fill(  
 child: Container(color: Colors.*black*.withOpacity(0.6)),  
 ),  
 // Content  
 Padding(  
 padding: const EdgeInsets.all(16.0),  
 child: SingleChildScrollView(  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 \_buildInputField(),  
 SizedBox(height: 20),  
 \_buildPredictButton(),  
 SizedBox(height: 20),  
 \_buildPredictionResults(),  
 ],  
 ),  
 ),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
  
 Widget \_buildInputField() {  
 return Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 Text(  
 'Enter Symptoms:',  
 style: TextStyle(fontSize: 18, fontWeight: FontWeight.*bold*, color: Colors.*white*),  
 ),  
 SizedBox(height: 10),  
 TextField(  
 controller: \_symptomsController,  
 decoration: InputDecoration(  
 hintText: 'e.g., shivering, headache, nausea',  
 hintStyle: TextStyle(color: Colors.*white70*),  
 filled: true,  
 fillColor: Colors.*white*.withOpacity(0.2),  
 border: OutlineInputBorder(  
 borderRadius: BorderRadius.circular(8.0),  
 ),  
 contentPadding: EdgeInsets.symmetric(horizontal: 16.0, vertical: 12.0),  
 ),  
 style: TextStyle(color: Colors.*white*),  
 maxLines: null, // Allows the TextField to grow with content  
 ),  
 ],  
 );  
 }  
  
 Widget \_buildPredictButton() {  
 return Center(  
 child: ElevatedButton(  
 onPressed: \_predictDisease,  
 child: Text('Predict'),  
 style: ElevatedButton.*styleFrom*(  
 padding: EdgeInsets.symmetric(horizontal: 50, vertical: 15),  
 textStyle: TextStyle(fontSize: 18),  
 ),  
 ),  
 );  
 }  
}

**ABOUT SCREEN. DART**

import 'package:flutter/material.dart';  
  
class AboutPage extends StatelessWidget {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 title: Text('About'),  
 backgroundColor: Colors.*greenAccent*,  
 ),  
 body: Stack(  
 children: [  
 // Background Image  
 Positioned.fill(  
 child: Image.asset(  
 'assets/images/background.jpg', // Ensure this image is in your assets folder  
 fit: BoxFit.cover,  
 ),  
 ),  
 // Semi-transparent overlay  
 Positioned.fill(  
 child: Container(  
 color: Colors.*black*.withOpacity(0.6),  
 ),  
 ),  
 // Content  
 Padding(  
 padding: const EdgeInsets.all(16.0),  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 Text(  
 'About the App',  
 style: TextStyle(  
 fontSize: 24,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*greenAccent*,  
 ),  
 ),  
 SizedBox(height: 20),  
 Text(  
 'This app predicts diseases based on symptoms provided by the user and offers medical recommendations. It aims to provide valuable insights and assistance to users seeking to understand their health conditions better.',  
 style: TextStyle(  
 fontSize: 16,  
 color: Colors.*white*,  
 ),  
 textAlign: TextAlign.justify,  
 ),  
 SizedBox(height: 30),  
 Text(  
 'Meet the Team:',  
 style: TextStyle(  
 fontSize: 22,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*greenAccent*,  
 ),  
 ),  
 SizedBox(height: 20),  
 Expanded(  
 child: ListView(  
 children: [  
 \_buildTeamMember(  
 name: 'Ebenezer Damptey',  
 role: 'Lead Developer',  
 imagePath: 'assets/images/Ebenezer.jpg',  
 ),  
 \_buildTeamMember(  
 name: 'Derrick Donkor',  
 role: 'UI/UX Designer',  
 imagePath: 'assets/images/Derrick.jpg',  
 ),  
 \_buildTeamMember(  
 name: 'Felix Dongbetigr',  
 role: 'Data Scientist',  
 imagePath: 'assets/images/Felix.jpg',  
 ),  
 \_buildTeamMember(  
 name: 'Amponsah Lucas',  
 role: 'UI/UX Designer',  
 imagePath: 'assets/images/Amponsah.jpg',  
 ),  
 \_buildTeamMember(  
 name: 'Sundong Wisdom Natonaah',  
 role: 'UI/UX Designer',  
 imagePath: 'assets/images/Sundong.jpg',  
 ),  
 ],  
 ),  
 ),  
 ],  
 ),  
 ),  
 ],  
 ),  
 );  
 }  
  
 Widget \_buildTeamMember({  
 required String name,  
 required String role,  
 required String imagePath,  
 }) {  
 return Padding(  
 padding: const EdgeInsets.symmetric(vertical: 8.0),  
 child: Row(  
 children: [  
 CircleAvatar(  
 radius: 30,  
 backgroundImage: AssetImage(imagePath),  
 ),  
 SizedBox(width: 10),  
 Expanded(  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 Text(  
 name,  
 style: TextStyle(  
 fontSize: 18,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*greenAccent*,  
 ),  
 ),  
 Text(  
 role,  
 style: TextStyle(  
 fontSize: 16,  
 color: Colors.*white*,  
 ),  
 ),  
 ],  
 ),  
 ),  
 ],  
 ),  
 );  
 }  
}

**CONTACT SCREEN. DART**

import 'package: flutter/material.dart';  
import 'package:url\_launcher/url\_launcher.dart';  
  
class ContactPage extends StatelessWidget {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 title: Text('Contact'),  
 backgroundColor: Colors.*greenAccent*,  
 ),  
 body: Stack(  
 children: [  
 // Background Image  
 Positioned.fill(  
 child: Image.asset(  
 'assets/images/background.jpg', // Ensure this image is in your assets folder  
 fit: BoxFit.cover,  
 ),  
 ),  
 // Semi-transparent overlay  
 Positioned.fill(  
 child: Container(  
 color: Colors.*black*.withOpacity(0.6),  
 ),  
 ),  
 // Content  
 Padding(  
 padding: const EdgeInsets.all(16.0),  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 Text(  
 'Contact the Developer',  
 style: TextStyle(  
 fontSize: 24,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*greenAccent*,  
 ),  
 ),  
 SizedBox(height: 20),  
 \_buildContactOption(  
 icon: Icons.*email*,  
 text: 'developer@example.com',  
 onTap: () async {  
 final Uri emailLaunchUri = Uri(  
 scheme: 'mailto',  
 path: 'developer@example.com',  
 );  
 if (await canLaunchUrl(emailLaunchUri)) {  
 await launchUrl(emailLaunchUri);  
 } else {  
 \_showError(context, 'Could not launch email app.');  
 }  
 },  
 ),  
 SizedBox(height: 10),  
 \_buildContactOption(  
 icon: Icons.*phone*,  
 text: '+233 57 028 2690',  
 onTap: () async {  
 final Uri phoneLaunchUri = Uri(  
 scheme: 'tel',  
 path: '+233 57 028 2690',  
 );  
 if (await canLaunchUrl(phoneLaunchUri)) {  
 await launchUrl(phoneLaunchUri);  
 } else {  
 \_showError(context, 'Could not launch phone app.');  
 }  
 },  
 ),  
 SizedBox(height: 30),  
 \_buildInputField('Your Name'),  
 SizedBox(height: 10),  
 \_buildInputField('Your Email'),  
 SizedBox(height: 10),  
 \_buildInputField('Your Message', maxLines: 4),  
 SizedBox(height: 20),  
 Center(  
 child: ElevatedButton.icon(  
 icon: Icon(Icons.*send*),  
 label: Text('Send Message'),  
 style: ElevatedButton.*styleFrom*(  
 backgroundColor: Colors.*greenAccent*,  
 padding:  
 EdgeInsets.symmetric(horizontal: 50, vertical: 15),  
 textStyle: TextStyle(fontSize: 18),  
 shape: RoundedRectangleBorder(  
 borderRadius: BorderRadius.circular(30),  
 ),  
 ),  
 onPressed: () {  
 // Handle the form submission or navigation  
 },  
 ),  
 ),  
 ],  
 ),  
 ),  
 ],  
 ),  
 );  
 }  
  
 Widget \_buildContactOption({required IconData icon, required String text, required Function() onTap}) {  
 return Row(  
 children: [  
 Icon(icon, color: Colors.*greenAccent*),  
 SizedBox(width: 10),  
 GestureDetector(  
 onTap: onTap,  
 child: Text(  
 text,  
 style: TextStyle(  
 fontSize: 18,  
 color: Colors.*blueAccent*,  
 decoration: TextDecoration.*underline*,  
 ),  
 ),  
 ),  
 ],  
 );  
 }  
  
 Widget \_buildInputField(String labelText, {int maxLines = 1}) {  
 return TextField(  
 maxLines: maxLines,  
 decoration: InputDecoration(  
 labelText: labelText,  
 labelStyle: TextStyle(color: Colors.*greenAccent*),  
 enabledBorder: OutlineInputBorder(  
 borderSide: BorderSide(color: Colors.*greenAccent*),  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 focusedBorder: OutlineInputBorder(  
 borderSide: BorderSide(color: Colors.*black*),  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 ),  
 style: TextStyle(color: Colors.*white*),  
 );  
 }  
  
 void \_showError(BuildContext context, String message) {  
 ScaffoldMessenger.*of*(context).showSnackBar(  
 SnackBar(content: Text(message)),  
 );  
 }  
}