HEART DISEASE PREDICTION

MAD PROJECT

Contents

[1. Introduction 2](#_Toc199586637)

[2. Project Background and Motivation 2](#_Toc199586638)

[3. Objectives and Scope 2](#_Toc199586639)

[4. Technology Stack Overview 3](#_Toc199586640)

[5. Backend Architecture FastAPI Framework 3](#_Toc199586641)

[Database: Firebase Firestore 3](#_Toc199586642)

[User Authentication and Authorization 3](#_Toc199586643)

[6. Machine Learning Models 4](#_Toc199586644)

[Logistic Regression 4](#_Toc199586645)

[Decision Tree Classifier 4](#_Toc199586646)

[Model Training and Storage 4](#_Toc199586647)

[7. API Design and Implementation 4](#_Toc199586648)

[8. Frontend Development 5](#_Toc199586649)

[Flutter Framework 5](#_Toc199586650)

[UI/UX Design and Flow 5](#_Toc199586651)

[9. Integration of Frontend and Backend 5](#_Toc199586652)

[10. User Management and Profiles 5](#_Toc199586653)

[11. Security and Validation 6](#_Toc199586654)

[12. Testing and Deployment 6](#_Toc199586655)

[13. Challenges and Solutions 6](#_Toc199586656)

[14. Future Work and Enhancements 7](#_Toc199586657)

[16. References 7](#_Toc199586658)

[17.Screenshots 8](#_Toc199586659)

[HEART-DISEASE-PREDICTION](https://github.com/roshiiiz/Heart-Disease-Prediction-App)

# 1. Introduction

Heart disease remains one of the leading causes of death worldwide. Early prediction and diagnosis can significantly improve treatment outcomes and reduce mortality. This project aims to develop a comprehensive Heart Disease Prediction application that leverages machine learning models to predict heart disease risk based on user inputs, while providing user authentication and a smooth interactive experience.

# 2. Project Background and Motivation

With the proliferation of digital health applications, integrating machine learning models with real-world user data provides powerful tools for proactive healthcare. This project is motivated by the need to create an accessible, easy-to-use app that empowers users with predictive insights into heart disease using clinically relevant factors.

# 3. Objectives and Scope

* To develop a backend service with secure user authentication using MySQL and FastAPI.
* To implement two machine learning models: Logistic Regression and Decision Tree.
* To build a Flutter-based mobile and desktop frontend for user interaction.
* To enable users to sign up, log in, select models, input health parameters, and view predictions.
* To design a scalable architecture supporting future model additions and user profile management.

# 4. Technology Stack Overview

|  |  |  |
| --- | --- | --- |
| Component | Technology/Framework | Purpose |
| Backend Framework | FastAPI | High-performance REST API |
| Database | |  | | --- | | Firebase Firestore |  |  | | --- | |  | | Data persistence for users and predictions |
| ML Models | scikit-learn (Logistic Regression, Decision Tree) | Model training and inference |
| Model Storage | joblib | Model serialization and loading |
| Frontend Framework | Flutter | Cross-platform mobile and desktop UI |
| Networking | HTTP + JSON | API communication between frontend and backend |
| IDE and Tools | Android Studio, VS Code | Development environments |

## 5. Backend Architecture FastAPI Framework

FastAPI was selected for its asynchronous capabilities, simplicity, and performance in building RESTful APIs. It provides automatic interactive API documentation and easy integration with Pydantic for data validation.

## Database: Firebase Firestore

Firebase Firestore is used to store user data, including authentication credentials and prediction logs. The app leverages Firebase for real-time data synchronization and scalability.

## User Authentication and Authorization

 **Registration:** Users can sign up with their email, password & Username.

 **Login:** Firebase Authentication handles the login process, verifying credentials.

 **Secure Storage:** Passwords are hashed before storage using Firebase’s secure methods.

 **Profile Management:** User profiles can be expanded to store user preferences and prediction history.

# 6. Machine Learning Models

## Logistic Regression

A widely used classification algorithm effective for binary outcomes. Trained on heart disease datasets, it predicts the likelihood of a heart condition.

## Decision Tree Classifier

A decision tree model offers interpretable rules based on input features, providing an alternative perspective to logistic regression.

## Model Training and Storage

* Models are trained using scikit-learn on the **heart.csv** dataset.
* After training, models are saved with **joblib** for quick loading during API calls.
* The backend loads these models at startup to serve prediction requests efficiently.

# 7. API Design and Implementation

* RESTful endpoints allow clients to interact with the backend.
* User-related endpoints: The /users endpoint allows for signup and login operations, leveraging Firebase Authentication.
* Prediction endpoint /predict/ accepts health parameter data in JSON and returns predictions from both ML models.
* Proper error handling and response codes ensure robustness.

# 8. Frontend Development

## Flutter Framework

Flutter was chosen to develop a responsive cross-platform app, supporting desktop and mobile.

## UI/UX Design and Flow

* A **Welcome Screen** introduces the app.
* A **Authentication Screen** Allows the user to Signup/Login.
* A **Model Selection Screen** allows users to pick Logistic Regression or Decision Tree.
* An **Input Screen** collects user health metrics with validation.
* A **Result Screen** displays prediction outcomes with probabilities.
* A **History Screen** displays Previous Predictions, which allows the user to view/delete their Predictions.

# 9. Integration of Frontend and Backend

 he Flutter frontend communicates with the FastAPI backend via HTTP requests using the http package.

 **JSON Responses:** The API sends back JSON responses, which the Flutter app parses and displays.

 **Local Testing:** For local development, the backend uses **127.0.0.1** (localhost) or **10.0.2.2** for Flutter emulators, or the machines IP for Physical Device

 **CORS Middleware:** FastAPI includes middleware to handle CORS (Cross-Origin Resource Sharing), enabling frontend-backend communication.

# 10. User Management and Profiles

* User profiles support login/signup flow (planned for expansion).
* Profiles allow storing prediction history, preferences.
* Database design supports extending user data with minimal changes.

# 11. Security and Validation

* Passwords hashed before storing.
* Input data validated both on client and server sides.
* HTTPS should be implemented for production.
* Proper error messages prevent sensitive information leaks.

# 12. Testing and Deployment

* Unit tests for backend API endpoints.
* Manual testing using Postman and Flutter emulators.
* Load testing prediction endpoint to ensure performance.
* Future plan for CI/CD pipeline integration and containerized deployment.

# 13. Challenges and Solutions

* **CORS Issues**: Solved by adding FastAPI CORS middleware.
* **Database Connection Errors**: Fixed connection strings and firewall rules.
* **Flutter Emulator Networking**: Used correct IP addresses (10.0.2.2 for emulator, 127.0.0.1 for desktop).
* **Model Serialization**: Used joblib to persist models.
* **UI Visibility**: Adjusted colors and styles to ensure readability.

# 14. Future Work and Enhancements

* Add user authentication with JWT tokens for secure sessions.
* Expand ML models to include Random Forest, Neural Networks.
* Add historical prediction data for users.
* Implement notification system for health tips.
* Integrate wearable data for real-time monitoring.
* Deploy backend to cloud service for scalability.
* Add multi-language support in UI.

15. Conclusion

This project successfully demonstrated building an end-to-end heart disease prediction system using modern technologies. It offers a user-friendly interface with robust backend and ML capabilities. With future enhancements, it can become a valuable tool for early diagnosis and personalized healthcare.

# 16. References

* scikit-learn documentation: https://scikit-learn.org/stable/
* FastAPI documentation: https://fastapi.tiangolo.com/
* Flutter documentation: https://flutter.dev/docs
* SQLAlchemy documentation: https://docs.sqlalchemy.org/
* MySQL official site: <https://www.mysql.com/>

# 17.Screenshots

A screenshot of a computer

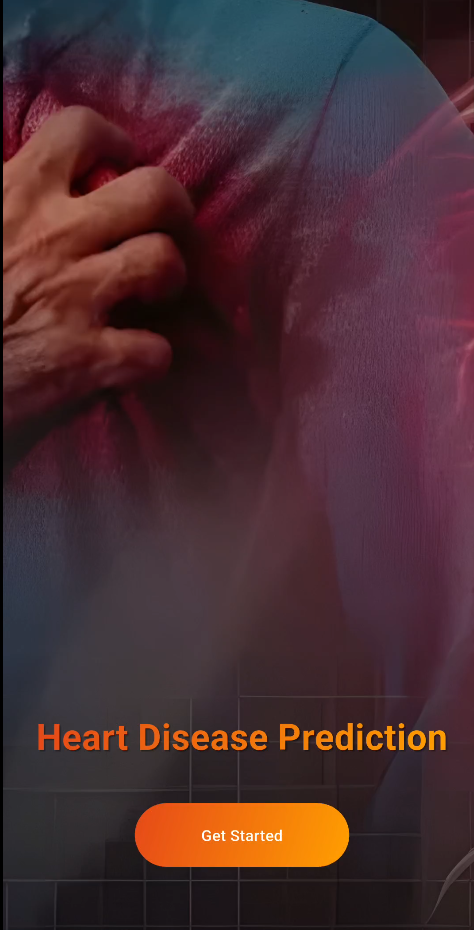
AI-generated content may be incorrect.

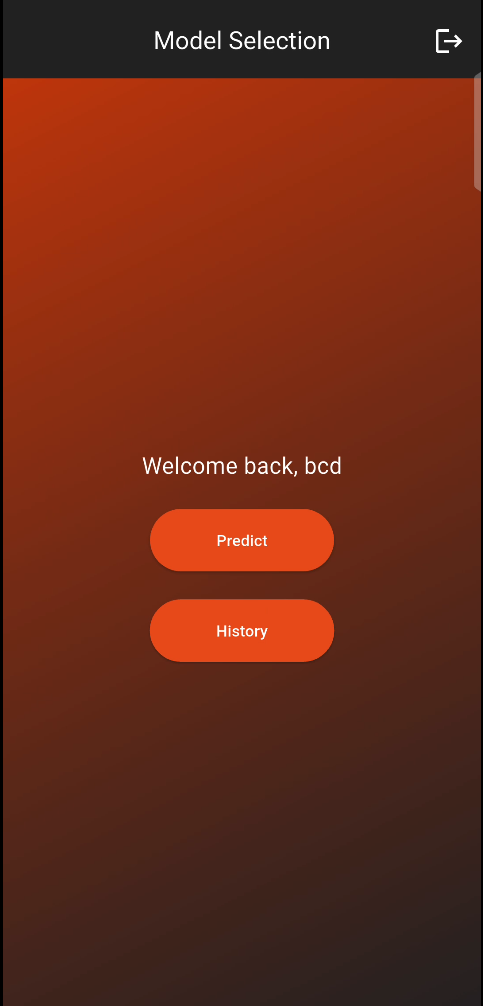
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a login screen

AI-generated content may be incorrect.A screenshot of a login form

AI-generated content may be incorrect.



A screen shot of a phone

AI-generated content may be incorrect.

