University of Constantine 2 – Abdelhamid Mehri

Department of Computer Science (IFA)

DiaPredict

AIoT-Based Application for Type 2 Diabetes Prediction

Prepared by:
Labed Ahmed Hchem
Louai Siradj Eddine Benkedda

Supervised by:
Dr. Ilham KITOUNI

Date: December 15th, 2025

1 Introduction

Diabetes mellitus is one of the most critical chronic diseases affecting millions of people worldwide. Among its types, **Type 2 Diabetes** is particularly concerning due to its strong relationship with lifestyle, hormonal changes, and metabolic factors. Pregnant women represent a high-risk group, as pregnancy induces physiological changes that may increase insulin resistance and blood glucose levels.

In this project, we focus on **Type 2 diabetes prediction for pregnant women** using the principles of **Artificial Intelligence of Things (AIoT)**. The proposed system, named **DiaPredict**, combines IoT sensors, cloud computing, machine learning, and data visualization to support early diabetes risk detection.

2 System Architecture

The DiaPredict application is designed following a multi-layer Artificial Intelligence of Things (AIoT) architecture. This layered design improves modularity, scalability, security, and ease of maintenance. Each layer is responsible for a specific set of functions, from data acquisition to intelligent decision-making and user interaction.

2.1 Perception Layer (Sensing Layer)

The perception layer is responsible for collecting raw physiological and environmental data from the user. In DiaPredict, this layer simulates medical sensors that measure key health indicators related to diabetes risk.

Layer Functions:

- Acquisition of real-time physiological data.
- Conversion of physical signals into digital data.
- Initial validation of sensor readings.

Technologies Used:

- Continuous Glucose Monitor (CGM) (simulated).
- Blood pressure, skin thickness, and insulin sensors.
- ESP32 microcontroller simulated using **Wokwi**.

2.2 Network Layer (Communication Layer)

The network layer ensures secure and reliable transmission of sensor data from the perception layer to cloud services. It acts as a bridge between edge devices and remote processing units.

Layer Functions:

- Transmission of sensor data in real time.

- Management of network connectivity and latency.
- Ensuring data integrity during transmission.

Technologies Used:

- **HTTP REST API** for data transmission.
- **MQTT** protocol for lightweight communication.
- Secure communication using **HTTPS / TLS**.

2.3 Cloud Layer (Data Processing and Storage)

The cloud layer acts as the central hub of the system. It stores incoming data, performs preprocessing, and makes the data available for analysis and visualization.

Layer Functions:

- Storage of real-time and historical sensor data.
- Data cleaning, normalization, and preprocessing.
- Data availability for AI inference and dashboards.

Technologies Used:

- **ThingSpeak** for IoT data ingestion.
- **Firebase Realtime Database** for user and prediction storage.
- **FastAPI (Python)** as the backend service.

2.4 Intelligence Layer (AI and Analytics)

This layer provides the intelligence of the DiaPredict system. It applies machine learning techniques to analyze health data and predict the risk of Type 2 diabetes.

Layer Functions:

- Feature extraction and selection.
- Diabetes risk prediction.
- Confidence score computation.

Technologies Used:

- **Decision Tree Classifier**.
- **Scikit-learn** for model training and inference.
- Python-based data processing pipelines.

2.5 Application Layer (Visualization and Interaction)

The application layer enables user interaction with the system. It provides visual representations of sensor data, predictions, and historical analytics.

Layer Functions:

- Real-time data visualization.
- Display of AI predictions and confidence levels.
- User interaction and navigation.

Technologies Used:

- **Next.js 14** with TypeScript.
- **Tailwind CSS** for responsive UI design.
- Arabic language support and modern glass-morphism UI.

2.6 Security Layer (Cross-Cutting Layer)

Security is implemented as a cross-cutting layer applied across all system components. This layer ensures confidentiality, integrity, and authorized access to sensitive medical data.

Layer Functions:

- User authentication and authorization.
- Secure data transmission and storage.
- Protection against unauthorized access.

Technologies Used:

- **JWT-based authentication**.
- Password hashing using **bcrypt**.
- Secure protocols (**HTTPS / TLS**).

3 Sensors

The DiaPredict system relies on a set of physiological sensors to collect medical data relevant to Type 2 diabetes risk assessment. These sensors capture real-time health indicators that are transmitted to the cloud for analysis and prediction.

3.1 Continuous Glucose Monitor (CGM)

The Continuous Glucose Monitor measures blood glucose levels at regular intervals. Glucose level is one of the most critical indicators for diabetes detection and monitoring. The CGM provides continuous readings that allow the system to detect abnormal glucose patterns over time.

3.2 Blood Pressure Sensor

The blood pressure sensor monitors systolic and diastolic blood pressure values. Hypertension is closely associated with diabetes risk, making blood pressure an important complementary feature for prediction.

3.3 Skin Thickness Sensor

Skin thickness measurements are used as an indirect indicator of body fat distribution. This feature is commonly included in diabetes datasets and contributes to improving the accuracy of the prediction model.

3.4 Insulin Level Sensor

The insulin sensor provides measurements related to insulin concentration in the blood. Abnormal insulin levels are a strong indicator of insulin resistance, which is a key factor in Type 2 diabetes.

3.5 Activity and Vital Sensors

Activity sensors monitor physical movement and basic vital signs such as heart rate. Physical activity level plays an important role in metabolic health and is therefore included in the overall risk assessment.

3.6 Sensor Simulation and Hardware Platform

During the prototyping phase, sensor data is simulated using an **ESP32** microcontroller environment. The ESP32 is emulated through the **Wokwi** platform, allowing realistic sensor data generation and seamless integration with IoT communication protocols before deployment on real hardware.

4 Artificial Intelligence Model

The intelligent component of DiaPredict is based on supervised machine learning techniques applied to medical data in order to predict the risk of Type 2 diabetes. Several models were evaluated to identify the most suitable algorithm in terms of performance and interpretability.

4.1 Dataset and Data Preparation

The experiments were conducted using the `diabetes.csv` dataset, which contains medical attributes commonly associated with diabetes diagnosis. The dataset was split into training (80%) and testing (20%) subsets. Feature scaling was applied using the **StandardScaler** to ensure numerical stability and improve model performance.

4.2 Evaluated Machine Learning Models

Five machine learning models were trained and evaluated for diabetes prediction:

- Logistic Regression
- Random Forest
- Support Vector Machine (SVM)
- XGBoost
- Decision Tree

Each model was assessed using accuracy and ROC-AUC metrics, along with confusion matrices and classification reports.

4.3 Model Comparison and Selection

A comparative analysis was performed to evaluate model performance. Table-based and graphical comparisons showed that the **Decision Tree classifier** achieved the best overall performance.

- Accuracy: **79.22%**
- ROC-AUC score: **0.8243**

Other models, such as Logistic Regression and XGBoost, showed competitive results but with lower overall accuracy. The Decision Tree was therefore selected due to its superior performance and high interpretability, which is crucial in medical decision-support systems.

4.4 Selected Model Configuration

The selected Decision Tree model was optimized using the following hyperparameters:

- Maximum tree depth: `max_depth = 5`
- Minimum samples per split: `min_samples_split = 19`

These parameters allowed the model to balance prediction accuracy and generalization while avoiding overfitting.

4.5 Model Deployment

After training and evaluation, the Decision Tree model was saved along with its metadata, including accuracy, ROC-AUC score, hyperparameters, and training date. The model is deployed within the DiaPredict backend to perform real-time diabetes risk prediction based on incoming IoT sensor data.

5 Dashboard Application

5.1 Frontend Architecture and Implementation

The DiaPredict dashboard frontend is implemented as a modern web application using the **Next.js** framework with **TypeScript** and **Tailwind CSS**. This architecture enables a modular, scalable, and maintainable user interface for real-time IoT data visualization and diabetes risk prediction.

5.1.1 Framework and Configuration

The frontend follows the Next.js App Router architecture. Core configuration files, such as `next.config.js`, `tsconfig.json`, and `package.json`, provide TypeScript support and application-level settings. Styling is managed using **Tailwind CSS**, configured through `tailwind.config.js` and `postcss.config.js`, enabling responsive design and consistent visual appearance.

5.1.2 Application Pages

The application is structured into several pages to support user navigation and functionality:

- **Landing Page**: Serves as the entry point of the application and presents the system features.

Figure 5.1: Landing Page of the DiaPredict Application

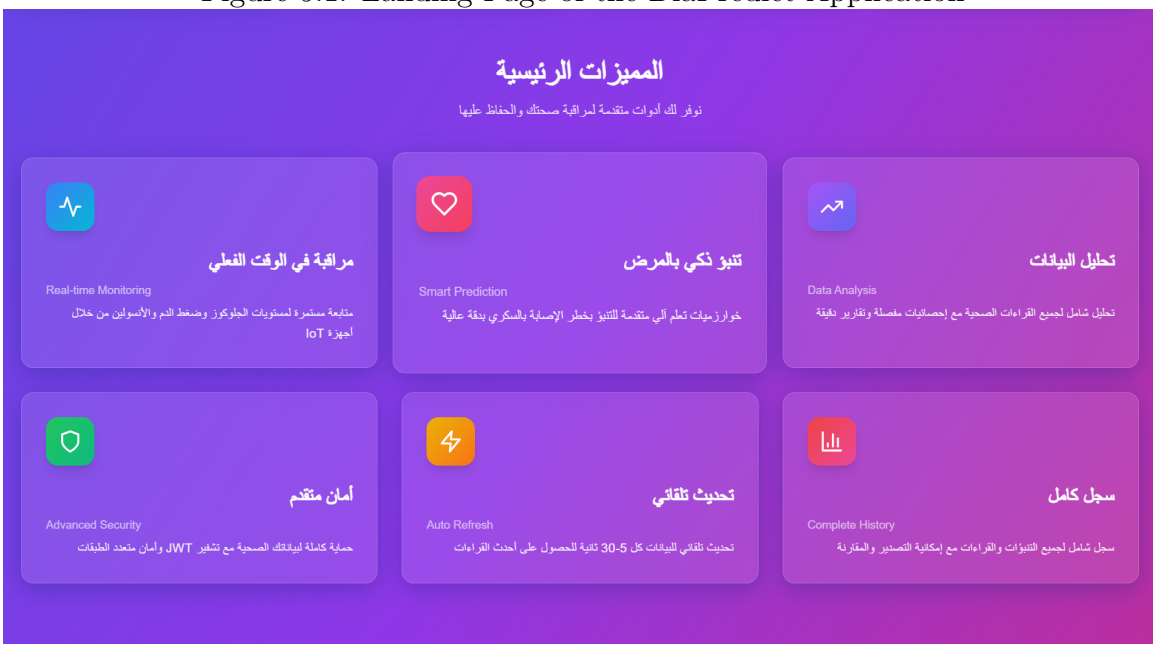
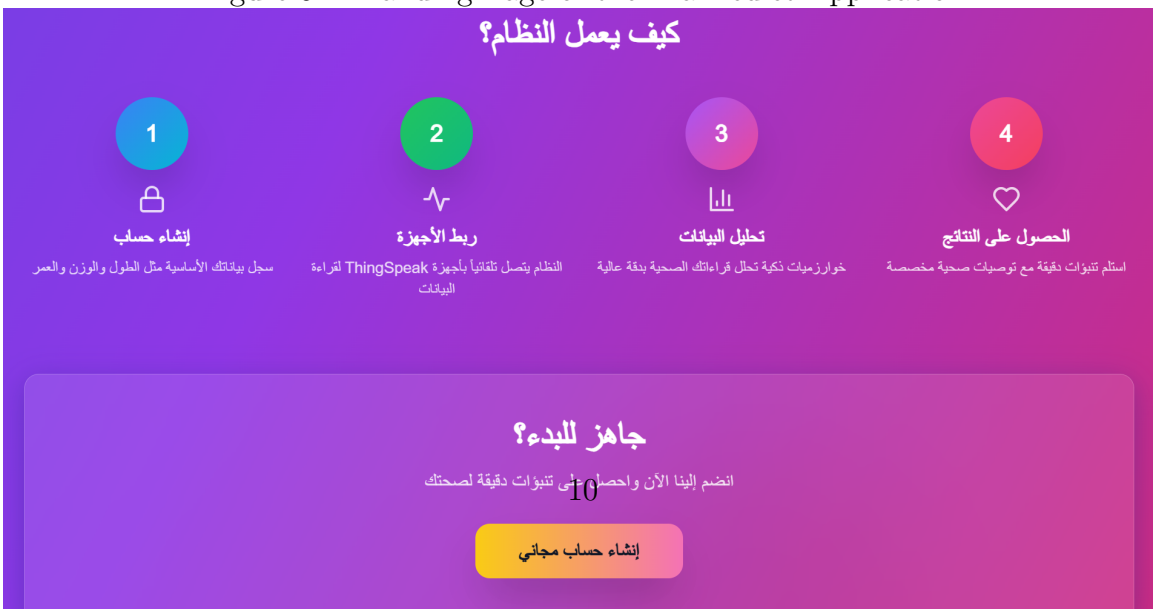


Figure 5.2: Landing Page of the DiaPredict Application



- **Dashboard Page**: Displays real-time IoT sensor data, prediction results, and analytics.

Figure 5.4: Dashboard of the DiaPredict Application

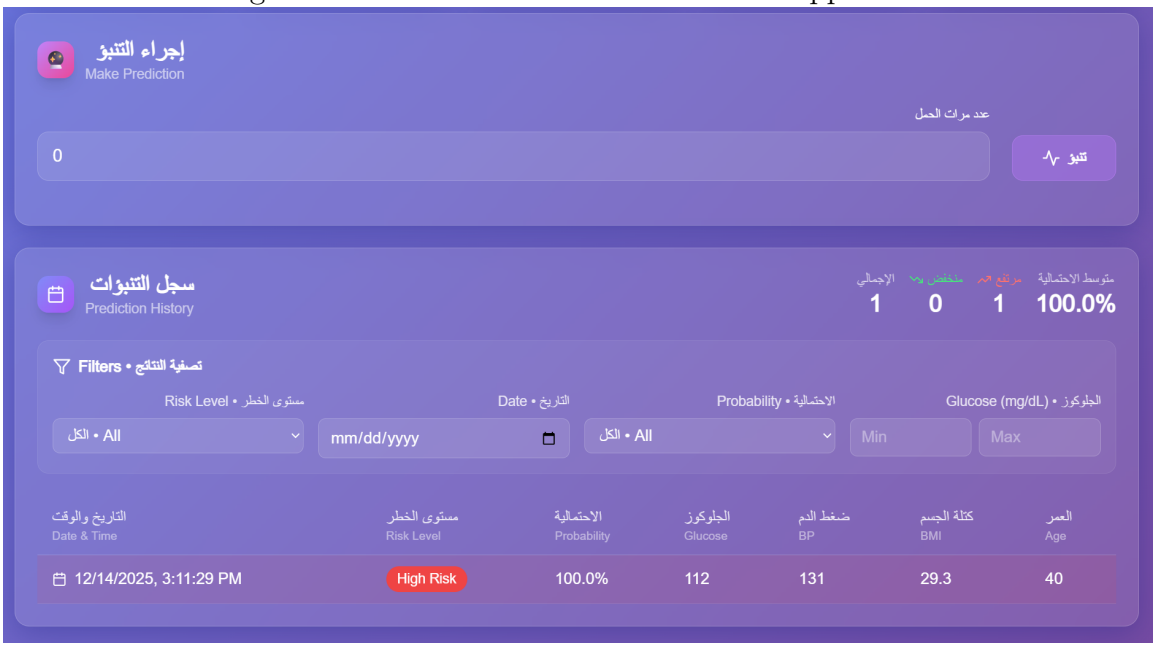


Figure 5.5: Dashboard of the DiaPredict Application

- **Authentication Pages**: Includes login and signup pages that control secure access to the system.

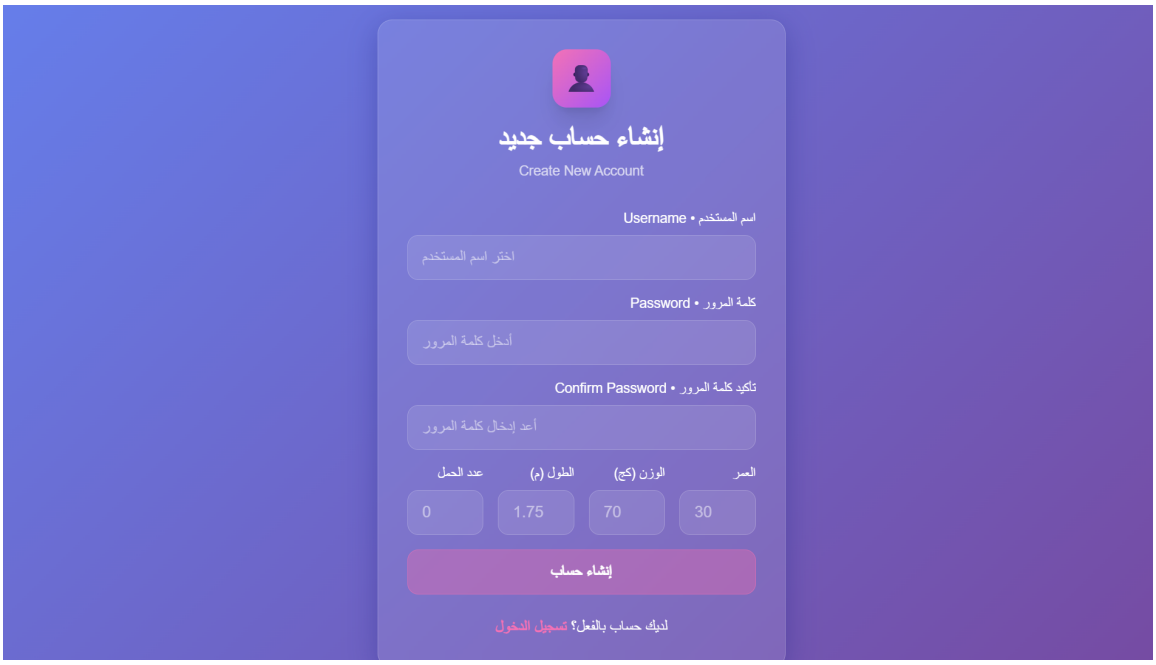


Figure 5.6: Sign Up Page of the DiaPredict Application

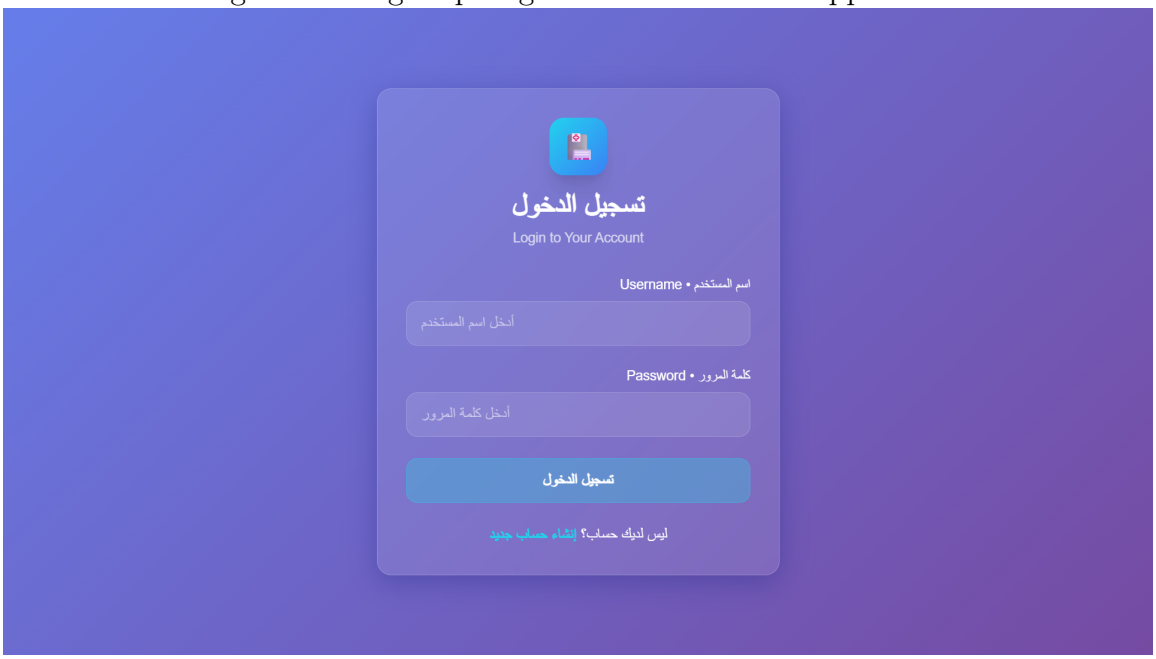


Figure 5.7: Sign In Page of the DiaPredict Application

This separation improves usability and enforces access control for protected resources.

5.1.3 Reusable Components

The frontend is built using reusable React components to enhance maintainability:

- **Sensor Cards**: Visualize real-time sensor readings retrieved from the ThingSpeak platform.

- **Prediction Panel**: Displays diabetes risk predictions generated by the deployed machine learning model.
- **History Table**: Presents historical sensor data and prediction results in a structured tabular format.

These components allow efficient rendering and dynamic updates of real-time data.

5.1.4 Data Access and Utilities

Communication with external services is handled through utility modules. API functions manage data fetching from the ThingSpeak platform and backend services, while authentication utilities control user sessions and access permissions. TypeScript type definitions are used to ensure data consistency and reduce runtime errors.

5.1.5 Deployment and Build Process

The frontend application is production-ready and includes compiled build artifacts generated by Next.js. The build process produces optimized static and server-rendered assets, allowing efficient deployment and high performance in real-world usage.

6 Security Mechanisms

Given the sensitive nature of medical and personal health data, DiaPredict integrates multiple security mechanisms across both backend and frontend components. The system follows standard best practices to ensure confidentiality, integrity, and controlled access to data.

6.1 Backend Security Mechanisms

The backend, implemented using a Python-based framework (FastAPI), is responsible for enforcing core security policies and protecting system resources.

6.1.1 Authentication and Authorization

User authentication and authorization are handled through dedicated authentication modules. Secure token-based mechanisms, such as JSON Web Tokens (JWT), are used to protect API endpoints and restrict access to authorized users only.

6.1.2 Data Validation and Protection

All incoming requests are validated using structured data models to ensure correctness and prevent injection attacks. Input and output validation mechanisms reduce the risk of malformed or malicious data reaching the database or machine learning model.

6.1.3 Secure Configuration Management

Sensitive configuration parameters, such as API keys and database credentials, are stored in environment variables rather than hardcoded in the source code. This approach minimizes the risk of secret leakage and supports secure deployment practices.

6.1.4 API Security

Backend APIs enforce security controls such as controlled CORS policies and secure communication with external services. Access to IoT platforms, such as ThingSpeak, is pro-

tected using API keys, ensuring that only authorized requests are accepted.

6.1.5 Model Security

The machine learning prediction service validates input data before inference to prevent invalid or adversarial inputs. The trained model is securely loaded from protected storage and exposed only through authenticated endpoints.

6.2 Frontend Security Mechanisms

The frontend application, developed using Next.js and TypeScript, applies additional security layers at the client level.

6.2.1 Client-Side Authentication

Authentication logic on the client side manages user sessions and access control. Tokens are handled securely, and automatic logout is triggered upon session expiration to prevent unauthorized access.

6.2.2 Secure API Communication

All communication between the frontend and backend is performed over secure HTTPS connections. API calls include authentication headers, and centralized error handling prevents information leakage.

6.2.3 Framework-Level Security

The Next.js framework provides built-in protection mechanisms, including automatic protection against common cross-site scripting (XSS) vulnerabilities. TypeScript enhances security by enforcing strict type checking and reducing runtime errors.

6.3 General Security Practices and Recommendations

In addition to implemented mechanisms, the system follows general security best practices:

- Use of HTTPS and TLS encryption for data in transit.
- Protection of sensitive data at rest.
- Regular dependency updates and vulnerability scanning.
- Logging and monitoring for abnormal activities.

These measures ensure that DiaPredict complies with fundamental security requirements expected in connected healthcare and AIoT systems.