

NAME: Muhammad Ibrahim Baba

REGISTRATION NUMBER: U18CO2017

**TOPIC: Development of an AI-Driven Nutritional Recommendations Application for
Diabetic Patients**

PROF. M. B. MUAZU
(Supervisor)

Date

CHAPTER ONE

INTRODUCTION

1.1 Background

Diabetes remains a chronic condition that has continued to affect millions of lives worldwide, Nigeria inclusive. According to the International Diabetes Federation (IDF), in 2019, around 463 million adults were living with diabetes, which is likely to soar to 700 million by 2045 (Federation 2019). Diet and nutrition are a core component in the management of diabetes, as changes in diet can significantly impact blood glucose levels. In Nigeria, where traditional diets are usually carbohydrate-heavy, managing blood sugar levels via diet becomes increasingly difficult. Proper dietary management is crucial in order to prevent complications of diabetes such as cardiovascular diseases, kidney failures, and neuropathies (Association 2020).

Over the past years, progress in artificial intelligence (AI) has gained huge hope in improving the healthcare sector by facilitating personal recommendations and supporting the automation of complex tasks. AI-powered applications perform data analysis and discern patterns within large datasets for insights that can go a long way in the management of chronic conditions, such as diabetes (Topol 2019). The plan of this project is to leverage AI technologies in developing software that can analyze Nigerian meals and make dietary recommendations according to the needs of Nigerian diabetic patients.

1.2 Project Motivation

The motivation for this project is, therefore, the increasing prevalence of diabetes and the need for an effective dietary management tool, especially that which is applicable to Nigerian foods. This makes it very hard for most diabetic patients in Nigeria to make the right food choices because

most nutritional information is complex, and traditional food habits are high in carbohydrates. Traditional methods of dietary management are mostly cumbersome, time-consuming, associated with poor compliance, and result in suboptimal health outcomes.

The advent of AI technology presents an opportunity to simplify and enhance the dietary management process for Nigerian diabetic patients. An AI-powered software, which our team develops with the ability to analyze Nigerian meals, might be a way to develop real-time dietary recommendations that would prove helpful in assisting patients to make informed choices regarding diet, hence improving quality of life and health outcomes. This project was additionally motivated by a desire to contribute to the growing field of AI in healthcare, especially in addressing unique health challenges faced in Nigeria.

1.3 Problem Statement

There are some challenges to dietary management, including lack of adequate, accessible, and reliable dietary guidance of Nigerian foods that may result in making inappropriate food choices by diabetic patients, thus worsening health conditions associated with diabetes. Most available methods of meal analysis and dietary recommendations are usually labor-intensive, time-consuming, and not personalized to common Nigerian dietary habits.

This identifies the problem of creating an intelligent, user-friendly, and accessible tool to help Nigerian diabetic patients make the right choice in terms of diet. Generally, there is a need for a software application that would analyze traditional Nigerian meals through image recognition and nutritional analysis, thereby giving personalized recommendations based on specified/diabetic patient requirements. The software should be able to analyze a wide variety of Nigerian foods, emphasizing the carbohydrate content and glycemic index in these foods, after which a resultant overall nutritional value of such foods can be considered.

1.4 Aim and Objectives

The aim of this project is to develop an AI-powered software application that analyzes meals and provides dietary recommendations for diabetic patients. The specific objectives of the project are as follows:

a. Design and Development of a User-Friendly Software Interface:

To design and develop an intuitive software interface that allows users to easily scan and upload images of their meals.

b. Data Collection and Dataset Development:

To gather and curate a comprehensive dataset of meal images and their corresponding nutritional information to support the AI analysis.

c. AI Models Development:

To implement and train AI models for accurate image recognition and nutritional analysis of various food items.

d. Development of the Project Back-End:

To create a robust back-end system that supports data processing, AI computations, and the recommendation engine.

e. Hosting and Cloud Infrastructure:

To establish a reliable hosting and cloud infrastructure that ensures the software's scalability, security, and accessibility for users.

1.5 Project Organization

The project is divided into five chapters. The first chapter introduces the project's background, motivation, problem definition, aim, and objectives. The second chapter covers the existing research on diabetes management, artificial intelligence in healthcare, and meal analysis technology. The third chapter explains the methodologies and strategies used to construct the software, such as system design, AI algorithms, and data collection. Chapter four describes the implementation process, including software development, testing procedures, and results. Finally, Chapter Five summarizes the findings, evaluates the project's ramifications, and makes recommendations for future research and development.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

Diabetes prevalence has risen dramatically over the world, needing appropriate management techniques to reduce its impact on people's health and quality of life. Maintaining a proper diet is an important element of diabetes care since it plays a key influence in blood glucose control. Artificial Intelligence (AI) breakthroughs in recent years have opened up new opportunities for personalized healthcare solutions, including novel ways to dietary analysis and suggestions.

This literature review looks at the present status of research and development in the field of Artificial Intelligence and diabetic dietary control. It investigates current diabetes management approaches and technology, the significance of AI in healthcare, and specific AI applications such as meal analysis and dietary advice. By combining various fields of research, this review attempts to provide a thorough knowledge of how AI-driven technologies might improve dietary control and diabetic patient outcomes.

2.2 Fundamental Concepts

The aim of this project is to develop an AI-powered software application using React Native to analyze meals and provide dietary recommendations for diabetic patients. This integration takes advantage of the strengths of both machine learning and cross-platform mobile development to offer a sophisticated, user-friendly solution for managing diabetes.

Integrating React Native with machine learning in this project enhances the functionality and accessibility of the application. It allows for real-time food analysis and personalized dietary recommendations, which are crucial for diabetic patients to manage their blood glucose levels effectively.

2.3 Diabetes

Diabetes mellitus is a chronic medical condition characterized by high levels of glucose (sugar) in the blood, resulting from the body's inability to produce or effectively use insulin. Insulin is a hormone produced by the pancreas that regulates blood sugar levels. Understanding the fundamental concepts of diabetes is crucial for developing applications aimed at managing this condition, such as AI-powered dietary recommendation systems for diabetic patients.

2.3.1 Types of Diabetes

Diabetes mellitus is classified into several types, primarily Type 1, Type 2, and gestational diabetes, each with distinct causes and management strategies.

a. Type 1 Diabetes

- i. **Autoimmune Condition:** Type 1 diabetes is an autoimmune disorder where the body's immune system attacks and destroys insulin-producing beta cells in the pancreas.
- ii. **Insulin Dependence:** Individuals with type 1 diabetes require lifelong insulin therapy to regulate their blood sugar levels.
- iii. **Onset and Symptoms:** It typically develops in childhood or adolescence but can occur at any age. Common symptoms include excessive thirst, frequent urination, weight loss, and fatigue.

b. Type 2 Diabetes

- i. **Insulin Resistance:** Type 2 diabetes is characterized by insulin resistance, where the body's cells do not respond effectively to insulin, and often a relative insulin deficiency.
- ii. **Risk Factors:** Major risk factors include obesity, physical inactivity, poor diet, age, and genetic predisposition.
- iii. **Management:** It can often be managed with lifestyle changes such as diet and exercise, but medications or insulin therapy may be needed as the condition progresses.

c. Gestational Diabetes

- i. **Pregnancy-Related:** This form of diabetes occurs during pregnancy and usually resolves after childbirth, but it increases the risk of developing type 2 diabetes later in life.
- ii. **Screening and Management:** Pregnant women are screened for gestational diabetes, and management includes dietary changes, physical activity, and sometimes insulin therapy.

2.3.2 Blood Glucose Monitoring

Effective blood glucose monitoring is essential for managing diabetes, enabling individuals to track their blood sugar levels and make informed decisions about diet, exercise, and medication.

Self-Monitoring of Blood Glucose (SMBG)

- i. **Importance:** Regular monitoring of blood glucose levels helps individuals with diabetes manage their condition by adjusting their diet, physical activity, and medication.
- ii. **Methods:** Common methods include finger-prick blood tests and continuous glucose monitors (CGMs) that provide real-time glucose readings.

Glycemic Index (GI)

The glycemic index measures how quickly foods cause blood glucose levels to rise. Foods with a high GI raise blood sugar levels rapidly, while those with a low GI have a slower, more gradual effect. The glycemic index (GI) is a ranking system for carbohydrates based on their immediate effect on blood glucose (blood sugar) levels. It compares the rise in blood glucose after consuming a specific food to the rise after consuming a standard reference food, such as glucose or white bread. Foods are assigned a GI value from 0 to 100, with higher values indicating a faster and more significant increase in blood glucose levels.

Classification of GI Values

- a. Low GI (55 or less): Foods that cause a slow, gradual rise in blood glucose levels. Examples include most fruits, vegetables, beans, and whole grains.
- b. Medium GI (56 to 69): Foods that cause a moderate increase in blood glucose levels. Examples include whole wheat products, sweet potatoes, and brown rice.
- c. High GI (70 and above): Foods that cause a rapid spike in blood glucose levels. Examples include white bread, white rice, and sugary foods

By selecting low-GI foods, patients can achieve better blood glucose control, reducing the risk of these complications. Low-GI diets have been shown to improve both glycemic control and lipid profiles in people with diabetes

2.3.3 Dietary Management for Diabetes

Dietary management for diabetes involves making informed food choices and planning meals to maintain stable blood glucose levels. This approach helps in managing the condition and improving overall health.

- i. **Nutrient-Rich Foods:** A balanced diet for diabetes management includes a variety of nutrient-rich foods, such as vegetables, fruits, whole grains, lean proteins, and healthy fats.
- ii. **Carbohydrate Counting:** Carbohydrate counting helps manage blood sugar levels by tracking the amount of carbohydrates consumed in each meal.

2.3.4 Web Scraping for Dataset Creation

Web scraping is a method used to extract large amounts of data from websites, which can then be used for various applications, including creating datasets for machine learning projects. In the context of this dietary management application, web scraping is employed to gather high-quality images of food items, particularly focusing on Nigerian local dishes. These images, along with their associated nutritional information, form an important part of the dataset used to train the object detection model.

2.3.5 Image Classification with Contrastive Language-Image Pre-training (CLIP)

The Contrastive Language-Image Pre-training (CLIP) model, developed by OpenAI, represents a significant advancement in the field of multimodal learning, particularly for tasks like image classification. CLIP's innovative approach involves training a model on a vast dataset of images paired with textual descriptions, enabling it to learn a wide range of visual concepts and their corresponding language representations. At its core, CLIP is trained using a contrastive learning technique where the model learns to match images with their corresponding textual descriptions. The model's architecture includes two main components: an image encoder and a text encoder. The image encoder processes the visual input, while the text encoder handles the associated textual

input. These encoders transform images and texts into embedding vectors within a shared multimodal space. During training, CLIP maximizes the cosine similarity between correct image-text pairs while minimizing it for incorrect pairs, allowing the model to establish strong associations between visual and linguistic data.

Zero-Shot Learning

One of the most powerful features of CLIP is its ability to perform zero-shot learning. Unlike traditional image classification models that require extensive labeled data for each task, CLIP can generalize to new tasks without additional training. This is achieved by leveraging the knowledge it gained from the diverse and extensive image-text pairs it was trained on. When applied to new datasets or classification tasks, CLIP can effectively categorize images by matching them with appropriate textual prompts, even if it has never encountered those specific categories before.

In the context of dietary management, CLIP's ability to classify images with minimal supervision is particularly valuable. For instance, in a system designed to monitor and manage the diets of diabetic patients, CLIP can be employed to identify various food items from images captured by the user. By matching these images with relevant textual descriptions, CLIP can provide accurate classifications that can be further used to retrieve nutritional information, such as caloric content, glycemic index, and other dietary factors crucial for managing diabetes.

2.3.6 Google Gemini Pro for LLM

Google Gemini Pro is an advanced large language model (LLM) developed by Google. It leverages state-of-the-art machine learning techniques to understand and generate human-like text based on the input it receives. Gemini Pro is designed to handle complex queries and provide accurate,

contextually relevant responses, making it a powerful tool for various applications, including natural language understanding, text generation, and conversational agents. Google Gemini Pro is built on the transformer architecture, which is the foundation of many modern LLMs. Google Gemini Pro is built on the transformer architecture, which is the foundation of many modern LLMs. The transformer model utilizes self-attention mechanisms to process input data efficiently, capturing long-range dependencies and contextual information. This architecture enables Gemini Pro to understand and generate coherent and contextually appropriate text. Transformers work by processing all words in a sentence simultaneously, which allows for more accurate understanding of context compared to previous models that processed words sequentially. One of the distinguishing features of Gemini Pro is its ability to handle multi-modal inputs, including text, images, and other data types. This capability allows the model to generate more comprehensive and contextually enriched responses, making it ideal for applications that require understanding and processing information from multiple sources. For example, it can analyze a picture and generate a descriptive text or understand text and generate relevant images.

2.3.7 Prompt Engineering

Prompt engineering is a critical aspect of leveraging large language models (LLMs) like Google Gemini Pro. It involves designing and refining input prompts to elicit the desired output from the model. Effective prompt engineering can significantly enhance the performance of AI systems by ensuring that the responses are accurate, relevant, and contextually appropriate. This section explores the principles, techniques, and best practices of prompt engineering, as well as its application in the dietary management application.

Principles of Prompt Engineering

- a. **Clarity and Specificity:** Prompts should be clear and specific to guide the model towards generating accurate and relevant responses. Ambiguous or vague prompts can lead to incorrect or irrelevant outputs. For instance, instead of asking, "What should I eat?" a more specific prompt would be, "What are some low-glycemic index foods suitable for a diabetic patient?"
- b. **Context Provision:** Providing context in the prompt helps the model understand the background and generate more appropriate responses. Context can include user information, previous interactions, or specific requirements. For example, "Based on my recent blood sugar readings, what adjustments should I make to my diet?"
- c. **Iterative Refinement:** Prompt engineering is an iterative process. Initial prompts are tested, and based on the outputs, they are refined to improve the accuracy and relevance of responses. This process continues until the desired performance is achieved.
- d. **Diversity in Prompts:** Using a variety of prompts during training and testing ensures that the model can handle a wide range of inputs. This helps in making the model robust and versatile in real-world applications.

Techniques of Prompt Engineering

- a. **Zero-shot Learning:** In zero-shot learning, the model is given a task it has not explicitly been trained on, using only the prompt for guidance. For example, "Explain the importance of portion control for diabetic patients" is a prompt that relies on the model's general understanding of the topic.
- b. **Few-shot Learning:** Few-shot learning involves providing the model with a few examples in the prompt to guide its response. This technique improves the model's performance by

showing it how to handle similar tasks. For example, "Here are examples of balanced meals for diabetics: 1) Grilled chicken with quinoa and steamed vegetables. 2) Baked salmon with brown rice and broccoli. Now, suggest another balanced meal."

- c. **Prompt Templates:** Creating templates for common queries can streamline the prompt engineering process. Templates provide a structured format that can be easily customized for different inputs. For instance, "What is the glycemic index of [food item]?" can be used to query the glycemic index of various foods by simply changing the food item.
- d. **Prompt Chaining:** In complex tasks, a series of prompts can be used to guide the model step-by-step towards the final output. This technique breaks down the task into smaller, manageable parts. For example, "List the ingredients for a healthy breakfast for diabetics. Next, provide a recipe using these ingredients."

2.3.7 LangChain for AI Integration

LangChain is a powerful framework designed to integrate various AI models, enabling seamless communication and coordination between these models and the application backend. It facilitates the deployment and management of large language models (LLMs) such as Google Gemini Pro, ensuring efficient utilization of AI capabilities in real-time applications.

Components of LangChain

LangChain provides standard, extendable interfaces and external integrations for various components useful for building with LLMs. Some components LangChain implements, while others rely on third-party integrations, or a mix of both.

- a. **Chat Models:** These are language models that use a sequence of messages as inputs and return chat messages as outputs. Chat models support the assignment of distinct roles to conversation messages, helping distinguish messages from the AI, users, and instructions such as system messages. LangChain integrates these models through third-party providers and supports various standardized parameters such as model name, temperature, timeout, max tokens, and API keys.
- b. **Multimodality:** Some chat models are multimodal, accepting images, audio, and even video as inputs. LangChain's abstraction supports these inputs, making it versatile in handling various data types.
- c. **LLMs (Language Models):** These models take a string as input and return a string. LangChain allows these models to take messages as input, giving them the same interface as Chat Models. This flexibility ensures that older models can be used effectively alongside newer chat models.
- d. **Messages:** LangChain supports different types of messages, such as `HumanMessage`, `AIMessage`, `SystemMessage`, `ToolMessage`, and legacy `FunctionMessage`. Each message type has specific properties and roles, enabling precise interaction and communication within the application.
- e. **Prompt Templates:** These help translate user input and parameters into instructions for a language model. LangChain supports various types of prompt templates, including `StringPromptTemplates`, `ChatPromptTemplates`, and `MessagesPlaceholder`, which allow for dynamic and flexible input formatting.

- f. **Example Selectors:** These are used to dynamically select and format examples into prompts, improving the performance of the language models by providing concrete examples of desired behavior.
- g. **Output Parsers:** These parsers transform the output of a model into a more suitable format for downstream tasks. LangChain supports various output parsers, ensuring that the generated data is structured and usable.

2.3.9 FastAPI for Backend Development

FastAPI is a modern, fast (high-performance) web framework for building APIs with Python 3.6+ based on standard Python type hints. It is designed to be easy to use and offers automatic interactive API documentation, which makes it a powerful tool for developing robust and scalable backend systems. FastAPI is known for its speed and efficiency, making it an ideal choice for developing the backend of a dietary management application where performance and reliability are crucial.

Key Features of FastAPI

- a. **High Performance:** FastAPI is built on top of Starlette for the web parts and Pydantic for the data parts. It uses asynchronous programming, which can handle many requests at once, making it extremely fast. This performance is comparable to frameworks like NodeJS and Go.
- b. **Automatic Interactive Documentation:** FastAPI generates interactive API documentation with Swagger UI and ReDoc. This feature is automatically included and allows developers to interact with the API directly through the documentation interface. This is particularly useful for testing and debugging.

- c. **Type Hints and Data Validation:** FastAPI leverages Python's type hints to provide data validation and serialization. This ensures that the data received and sent by the API adheres to the defined schema, reducing the likelihood of errors and improving code quality.
- d. **Dependency Injection:** FastAPI provides a robust dependency injection system. This allows developers to manage dependencies cleanly and efficiently, making the codebase more maintainable and scalable.
- e. **Security and Authentication:** FastAPI includes built-in tools for handling security, including OAuth2, JWT, and basic authentication. This simplifies the implementation of secure endpoints and user authentication processes.

2.3.10 PostgreSQL for database

PostgreSQL is an advanced, open-source relational database management system (RDBMS) known for its robustness, scalability, and compliance with SQL standards. It supports various data types, including JSON and arrays, and offers advanced indexing techniques like B-tree and GiST, optimizing query performance. PostgreSQL is fully ACID-compliant, ensuring reliable transaction processing and data integrity. Its Multi-Version Concurrency Control (MVCC) system allows multiple users to perform read and write operations simultaneously without conflicts. Extensibility is a key feature, allowing users to define custom data types, operators, and index methods. Security features include user authentication, data encryption, and access control mechanisms. In the dietary management application, PostgreSQL is used to store and manage data such as user information, dietary records, and nutritional information. The database schema includes tables for users, food entries, meals, and nutritional information, with relationships defined through foreign keys. Integration with FastAPI is achieved using SQLAlchemy, providing ORM capabilities to interact with the database. CRUD operations are implemented to manage data, ensuring data integrity.

through constraints like primary keys and unique indexes. Query optimization techniques, such as indexing, are employed to enhance performance. Hosting the database on Render ensures reliable and scalable infrastructure, while security measures protect data against unauthorized access. Overall, PostgreSQL's features make it an ideal choice for managing the application's data efficiently and securely.

2.3.11 Hosting: Render and AWS

Hosting is a critical aspect of deploying and maintaining a web application, ensuring it is accessible, reliable, and scalable. For the dietary management application, Render and AWS are used to provide the necessary infrastructure for hosting the backend services and AI components.

Render for Backend Hosting

Render is a modern cloud platform designed to simplify the deployment and management of web applications. It provides a fully managed service with features that cater to the needs of modern web applications, including automated deployments, scalability, and integrated HTTPS.

- a. **Ease of Use:** Render offers a user-friendly interface and automated deployment pipelines that simplify the process of deploying applications. Developers can push code to a Git repository, and Render automatically builds and deploys the application.
- b. **Scalability:** Render automatically scales applications based on traffic, ensuring that the application can handle varying loads without manual intervention. This is particularly useful for applications with fluctuating usage patterns.
- c. **Managed Databases:** Render provides managed PostgreSQL databases, which are integrated with the backend services. This ensures reliable and efficient database management, including automated backups and monitoring.

- d. **HTTPS and Security:** Render includes built-in HTTPS for all applications, enhancing security by encrypting data transmitted between the client and server. It also offers features such as DDoS protection and network isolation to secure the application from external threats.

AWS for AI and Storage

Amazon Web Services (AWS) is a comprehensive cloud platform offering a wide range of services for computing, storage, and AI. For the dietary management application, AWS is used to host the AI components and manage storage needs.

- a. **Amazon S3 for Storage:** AWS S3 (Simple Storage Service) is utilized for storing large datasets, including food images and other static content. S3 provides scalable and durable storage, ensuring data availability and integrity. It supports various storage classes, allowing cost optimization based on access patterns.
- b. **Amazon EC2 for AI Hosting:** The AI components, including the Google Gemini Pro large language model, are hosted on Amazon EC2 (Elastic Compute Cloud). EC2 offers a range of instance types tailored to different workloads, providing the flexibility to choose instances with the appropriate computational power for training and inference tasks.
- c. **AWS Lambda for Serverless Computing:** AWS Lambda is used for running code in response to events, enabling a serverless architecture for specific functionalities. Lambda functions can be triggered by various events, such as HTTP requests or changes in S3 buckets, providing a scalable and cost-effective way to handle specific tasks.
- d. **Security and Compliance:** AWS offers robust security features, including Identity and Access Management (IAM) for granular control over permissions, encryption of data at

rest and in transit, and compliance with various industry standards. These features ensure that the AI components and data are secure and meet regulatory requirements.

2.4 React Native Fundamentals

React Native is an open-source mobile application framework developed by Facebook. It enables developers to build natively-rendered mobile apps for iOS and Android using JavaScript and React. React Native combines the best parts of native development with React, a best-in-class JavaScript library for building user interfaces.

2.4.1 Key Features and Benefits

- a. Cross-Platform Development: Write once, run anywhere. React Native allows code sharing across iOS and Android platforms, reducing development time and cost.
- b. Hot Reloading: Enables developers to see changes in real-time without recompiling the entire application.
- c. Native Modules and Libraries: Access native platform features using JavaScript and third-party libraries.
- d. Strong Community Support: A large, active community contributes to a rich ecosystem of plugins and tools.

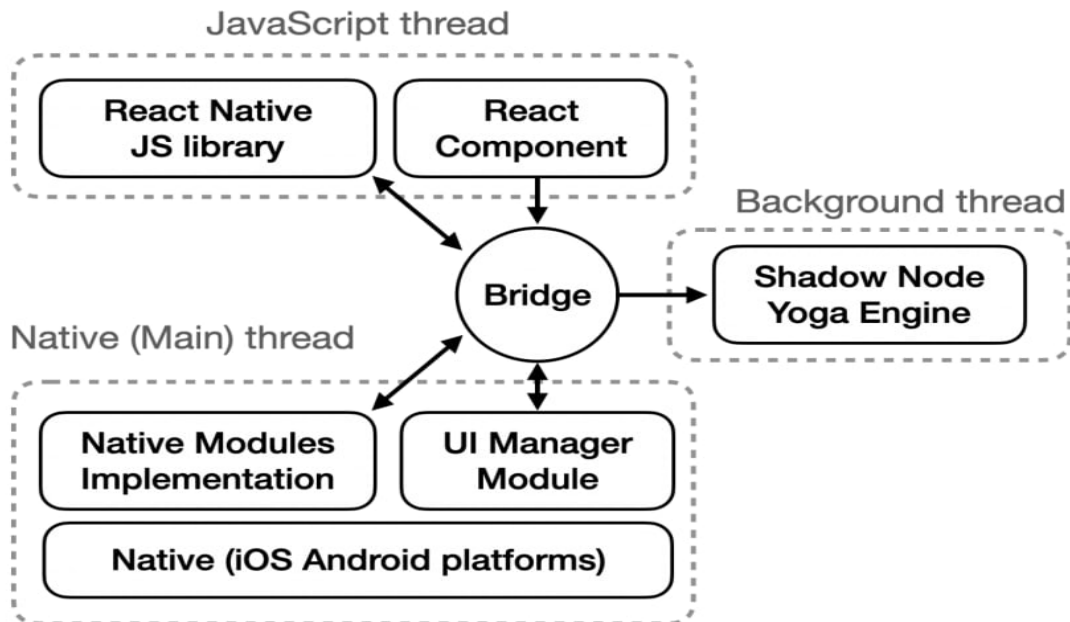
2.4.2 React Native Architecture

React Native's architecture includes three main layers: the JavaScript thread, the native thread, and the bridge between them. Figure 2.1 below shows the graphical representation of this architecture.

- a. JavaScript Thread: Runs the application logic written in JavaScript.
- b. Native Thread: Handles native UI components and platform-specific functionalities.

- c. Bridge: Facilitates communication between the JavaScript and native threads, enabling the JavaScript code to control native components.

Diagram: React Native Architecture



2.5

Figure 2. 1 React Native Architecture

Machine

Learning Basics

Machine learning (ML) is a subset of artificial intelligence (AI) that focuses on building systems that learn from data and improve their performance over time without being explicitly programmed.

2.5.1 Types of Machine Learning

- Supervised Learning:** The model is trained on labeled data. Example: Classifying food items based on images.
- Unsupervised Learning:** The model finds patterns and relationships in unlabeled data. Example: Grouping similar food items together.
- Reinforcement Learning:** The model learns by interacting with the environment and receiving feedback. Example: Optimizing dietary recommendations based on user feedback.

2.5.2 Common Machine Learning Algorithms

- a. Decision Trees: Used for classification and regression tasks.
- b. Random Forests: An ensemble method that improves the accuracy by combining multiple decision trees.
- c. Naive Bayes: A probabilistic classifier based on Bayes' theorem.
- d. Neural Networks: Models that mimic the human brain, suitable for complex tasks like image recognition.

2.6 Integrating Machine Learning with React Native

Integrating machine learning with React Native enables the creation of intelligent mobile applications that leverage advanced analytics and real-time data processing. This section outlines the steps to incorporate machine learning models into React Native apps, including loading models, handling input data, performing inference, and displaying results, providing a seamless and efficient user experience.

2.6.1 Setting Up the React Native Project

Step 1: Initialize a React Native project using the command:

```
`npx react-native init GlucoBuddy`
```

Step 2: Install necessary dependencies and libraries:

```
`npm install @tensorflow/tfjs @tensorflow/tfjs-react-native`
```

2.6.2 Choosing the Right Machine Learning Library

Selecting the appropriate machine learning library is a critical decision that impacts the efficiency, performance, and scalability of a machine learning project. The choice of library depends on several factors, including the specific requirements of the project, the complexity of the models to

be used, the computational resources available, and the deployment environment. Here, some key considerations and popular libraries to help guide this decision were explored.

Project Requirements and Model Complexity

The first consideration when choosing a machine learning library is the specific requirements of the project. Different libraries offer varying levels of support for different types of models and algorithms. For example, if the project involves deep learning and requires advanced neural network architectures, libraries like TensorFlow and PyTorch are well-suited due to their extensive support for complex neural networks and deep learning frameworks. TensorFlow, developed by Google, is known for its flexibility and scalability, making it suitable for both research and production environments. PyTorch, developed by Facebook, is favored for its dynamic computational graph, which offers more flexibility and ease of use during the model development phase.

Ease of Use and Community Support

The ease of use of a library and the availability of community support are also crucial factors. Libraries with extensive documentation, tutorials, and active user communities can significantly reduce the learning curve and provide valuable resources when troubleshooting issues. For instance, Scikit-learn is renowned for its simplicity and ease of use, making it an excellent choice for beginners and for projects that require traditional machine learning algorithms like regression, classification, and clustering. Its comprehensive documentation and supportive community make it a go-to library for many machine learning practitioners.

Computational Resources and Performance

The computational resources available and the performance requirements of the project are another important consideration. TensorFlow and PyTorch both offer support for GPU acceleration, which is essential for training large-scale neural networks efficiently. TensorFlow's Tensor Processing Unit (TPU) support can further enhance performance for large-scale deployments. For projects with more modest resource requirements or for those that need to run on edge devices, TensorFlow Lite offers a lightweight solution optimized for mobile and embedded devices, allowing for efficient on-device machine learning.

Deployment Environment

The deployment environment is also a critical factor in selecting a machine learning library. For web-based applications, TensorFlow.js allows models trained in TensorFlow to be deployed and run in the browser, leveraging JavaScript. This is particularly useful for creating interactive, web-based machine learning applications. For mobile applications, TensorFlow Lite and ONNX (Open Neural Network Exchange) provide solutions for deploying models on Android and iOS devices, ensuring that the models run efficiently on these platforms.

Interoperability and Flexibility

Interoperability and flexibility of the library also play a significant role. ONNX, for example, provides a standardized format that allows models to be transferred between different frameworks. This is particularly useful for projects that may need to switch between different libraries during different phases of development or deployment. Additionally, libraries like Keras, which acts as a high-level API for TensorFlow, offer ease of use while still providing the flexibility to work with lower-level TensorFlow operations when needed.

In summary, choosing the right machine learning library involves a careful consideration of the project's requirements, model complexity, ease of use, community support, computational resources, performance needs, and the deployment environment. TensorFlow and PyTorch are excellent choices for deep learning projects requiring high performance and flexibility, while Scikit-learn is ideal for traditional machine learning tasks due to its simplicity and robustness. TensorFlow Lite and TensorFlow.js extend the capabilities of TensorFlow to mobile and web applications, respectively, ensuring that machine learning models can be deployed efficiently across various platforms.

2.6.3 Model Training and Conversion

Model training and conversion are fundamental processes in the development and deployment of machine learning models, particularly in the context of integrating these models into various platforms and applications. Training a machine learning model involves several steps, starting with data preparation. This includes collecting relevant datasets, performing necessary preprocessing steps such as normalization, data augmentation, and splitting the dataset into training, validation, and test sets. These steps ensure that the data is clean and suitable for training, helping the model to generalize well to new, unseen data.

Once the data is prepared, the next step is designing the model architecture. For example, convolutional neural networks (CNNs) are commonly used for image-related tasks due to their ability to capture spatial hierarchies in data. The model is then compiled by specifying the optimizer (e.g., Adam), loss function (e.g., categorical cross-entropy for classification tasks), and evaluation metrics (e.g., accuracy). The training process involves iteratively feeding the training data into the model and adjusting the model weights to minimize the loss function. This is typically

done over multiple epochs, with the model's performance being monitored on the validation set to prevent overfitting and ensure generalization.

After the model is trained and validated, it needs to be converted into a format that can be deployed in the target environment. This conversion allows the model to be executed in JavaScript environments, such as web browsers or mobile applications using frameworks like React Native. The conversion process ensures that the model retains its learned parameters and structure, enabling it to perform inference efficiently in the new environment.

In the case of models trained with PyTorch, an intermediate step involves converting the model to the Open Neural Network Exchange (ONNX) format. ONNX serves as an intermediate representation that facilitates interoperability between different machine learning frameworks. Once in ONNX format, the model can be further converted to a format compatible with the deployment platform, such as TensorFlow.js for web and mobile applications.

Integrating the converted model into the application involves loading the model using the appropriate library (e.g., TensorFlow.js for JavaScript environments) and implementing functions to perform inference. This process ensures that the model can process input data and generate predictions in real-time, enabling applications to leverage advanced machine learning capabilities seamlessly.

Overall, model training and conversion are critical steps that bridge the gap between developing machine learning models and deploying them in practical, real-world applications. These processes ensure that models are not only trained effectively but also optimized and converted for efficient execution in their target environments, thereby enhancing the functionality and user experience of the applications they power.

2.7 Data Collection and Preparation

Data collection and preparation are foundational steps in any machine learning project, as they ensure that the data used to train models is accurate, clean, and relevant. These steps involve gathering data from various sources, performing necessary preprocessing, and preparing the data in a format suitable for model training.

2.7.1 Data Collection

Data collection involves identifying sources from which data can be gathered. Primary data is collected directly from the source through methods such as surveys, experiments, or observations, tailored to the specific needs of the project. Secondary data is obtained from existing sources like public datasets, research studies, databases, and APIs, with examples including the UCI Machine Learning Repository, Kaggle datasets, and government databases. This combination of primary and secondary data ensures a comprehensive and relevant dataset for analysis.

2.7.2 Gathering Data

Data can be gathered using various methods to support machine learning projects. Web scraping involves using automated tools and scripts to extract data from websites, with Python libraries like BeautifulSoup and Scrapy commonly employed for this purpose. APIs provide a programmatic way to access data from platforms such as social media, financial services, and weather providers. Additionally, data collection can be facilitated through sensors and IoT devices, which are particularly useful in fields like healthcare, agriculture, and smart cities, allowing for real-time data acquisition and monitoring.

2.7.3 Ensuring Data Quality

Ensuring data quality involves several key aspects: relevance, accuracy, completeness, and timeliness. Relevance ensures that the collected data is pertinent to the problem being addressed,

while accuracy verifies that the data is free from errors. Completeness checks that the dataset has no missing values or gaps, and timeliness ensures that the data is up-to-date and relevant to the current context. These steps are crucial to maintain the integrity and reliability of the dataset, directly impacting the effectiveness of the machine learning models built from it.

2.7.4 Data Preparation

Data preparation involves several critical steps to ensure the dataset is suitable for model training. Data Cleaning includes handling missing values through techniques like imputation (replacing with mean, median, or mode), deletion, or using algorithms that manage missing data, removing duplicates to prevent skewed results, and correcting errors or inconsistencies such as outliers. Data Transformation entails normalization, scaling data to a specific range (e.g., 0 to 1) using techniques like min-max scaling and z-score normalization, encoding categorical data into numerical formats via one-hot or label encoding, and feature engineering to create new features from existing data, improving model performance with polynomial features, interaction terms, or domain-specific transformations. Data Splitting involves dividing the dataset into training (70-80%), validation, and test sets to train the model, tune parameters, and evaluate performance, respectively. Data Augmentation uses techniques like rotation, flipping, and cropping for image data, and synonym replacement or back translation for text data to artificially expand the training dataset. Tools like Pandas, NumPy, Scikit-learn, TensorFlow, and PyTorch facilitate these processes by providing robust frameworks for data manipulation, numerical operations, and preprocessing.

2.8 Feature Selection and Engineering

Feature selection and engineering are critical steps in the machine learning pipeline, directly impacting the performance and interpretability of the models. These processes involve identifying

the most relevant features from the dataset and creating new features that can enhance the model's predictive power.

2.8.1 Feature Selection

Feature selection aims to reduce the dimensionality of the dataset by selecting only the most significant features, which helps to improve model performance by eliminating irrelevant or redundant features. This process reduces the risk of overfitting, making the model more generalizable, and decreases computational cost and complexity. Additionally, feature selection enhances the interpretability of the model by focusing on the most impactful data.

2.8.2 Techniques for Feature Selection

Feature selection involves various techniques to identify the most relevant features for a machine learning model. Filter methods evaluate the relevance of features based on statistical tests independent of any machine learning algorithm, using measures such as the correlation coefficient, chi-square test, and ANOVA (Analysis of Variance) to assess feature importance. Wrapper methods evaluate subsets of features based on their performance with a specific machine learning algorithm, employing techniques like forward selection, backward elimination, and recursive feature elimination (RFE) to iteratively refine the feature set. Embedded methods integrate feature selection into the model training process itself, with examples including Lasso regression, which uses L1 regularization to shrink coefficients to zero, and tree-based methods such as decision trees and ensemble methods like Random Forests and Gradient Boosting, which naturally rank features by importance during training. These techniques collectively enhance model performance by selecting the most significant features and eliminating redundant or irrelevant ones.

2.8.3 Evaluating Feature Importance

Evaluating feature importance helps determine which features contribute most to the model's predictions. In linear models, the absolute value of the coefficients indicates feature importance, with higher magnitudes signifying more influence. Tree-based methods, such as decision trees and random forests, provide feature importance scores based on how often and effectively features are used to split nodes. Permutation importance measures the change in model performance when a feature's values are randomly shuffled, indicating the feature's contribution by observing the resulting decrease in accuracy. These methods collectively help in understanding and interpreting the model's behavior, ensuring that the most relevant features are identified and utilized.

2.8.4 Feature Engineering

Feature engineering involves creating new features or transforming existing ones to improve model performance. This process enhances the predictive power of the model, makes it more robust to variations in the data, and incorporates domain knowledge into the model. Techniques for feature engineering include transformation, where mathematical functions such as logarithms, square roots, or exponential transformations are applied to stabilize variance or make relationships more linear. Interaction features involve creating new features by combining existing ones, capturing interactions through multiplication or addition. Polynomial features are generated by raising existing features to a power. Binning converts continuous features into categorical ones by dividing them into intervals. Encoding categorical variables transforms them into numerical format using methods like one-hot encoding, label encoding, or target encoding. Additionally, extracting meaningful components from date and time features, such as day of the week, month, or hour of the day, can provide valuable insights.

2.8.5 Handling Missing Values

Handling missing values is crucial for maintaining data integrity and improving model performance. Imputation involves replacing missing values with statistical measures such as the mean, median, or mode, ensuring that the dataset remains complete and consistent. Predictive imputation goes further by using machine learning models to predict and fill in missing values based on other features in the dataset, which enhances the accuracy and robustness of the data. Both techniques are essential for effectively managing incomplete data and ensuring that machine learning models are trained on reliable datasets.

2.8.6 Scaling and Normalization

Scaling and normalization are essential preprocessing steps in machine learning that ensure features contribute equally to the model's performance. Standardization scales features to have a mean of zero and a standard deviation of one, which is crucial for algorithms that assume normally distributed data. Min-Max Scaling transforms features to a fixed range, typically $[0, 1]$, which is beneficial for algorithms that require bounded input values. Robust Scaling uses statistics that are robust to outliers, such as the median and interquartile range, ensuring that the scaling process is not unduly influenced by extreme values. These techniques enhance the model's convergence speed and accuracy by ensuring all features are on a comparable scale.

2.8.7 Tools and Libraries

- a. Pandas: Provides powerful data manipulation and analysis capabilities, including functions for handling missing values, scaling, and encoding.
- b. Scikit-learn: Offers a range of preprocessing utilities and feature selection methods.
- c. Feature-engine: A Python library specifically designed for feature engineering, providing a suite of transformers to automate common feature engineering tasks.

2.9 Developing the Machine Learning Model

Developing the machine learning model involves selecting appropriate algorithms, training the model with data, and fine-tuning it to ensure high accuracy and generalizability. This process is crucial for creating models that can make reliable predictions and perform well in real-world scenarios.

2.9.1 Model Training and Evaluation

Model training and evaluation are critical stages in the machine learning pipeline. During the training phase, the model is trained using a designated training dataset, allowing it to learn patterns and make predictions based on input data. This involves feeding the training data into the model and iteratively adjusting its parameters to minimize the error, typically measured by a loss function. Once trained, the model's performance is assessed using various evaluation metrics. Precision, recall, F-measure, and accuracy are commonly used metrics to gauge how well the model predicts and classifies new data. Precision measures the accuracy of the positive predictions, recall indicates the ability of the model to find all relevant instances, the F-measure balances precision and recall, and accuracy provides an overall correctness of the model. These metrics help in understanding the strengths and weaknesses of the model, guiding further improvements and tuning.

2.9.2 Model Optimization and Fine-Tuning

Model optimization and fine-tuning are essential for enhancing the performance of machine learning models. Optimization techniques involve adjusting hyperparameters, employing regularization methods, and using cross-validation to ensure robust performance. Fine-tuning further improves the model by incorporating additional data or modifying the model architecture

to better capture underlying patterns. Figure 2.2 below illustrates the entire machine learning process, highlighting these crucial steps.

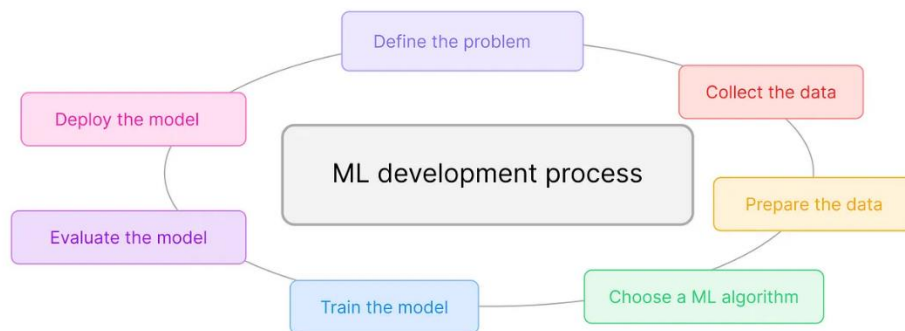


Figure 2. 2 Machine Learning Model Training Process

2.10 User Interface and Experience

Designing the User Interface involves following best practices to create intuitive and user-friendly interfaces, utilizing components like buttons, lists, and modals for user interaction. Ensuring accessibility by adhering to guidelines and using appropriate ARIA roles is crucial for making the app usable for individuals with disabilities. User Interaction and Feedback is essential for continuously improving the app's functionality and user experience by capturing user interactions and feedback. Implementing features such as notifications, user profiles, and feedback forms helps engage users and gather valuable insights for enhancements.

2.11 Security and Privacy Considerations

Security and privacy considerations are critical components in the development of any application, especially those handling sensitive data such as personal health information. This section addresses

the best practices and techniques for ensuring data protection, user privacy, and regulatory compliance. By implementing robust security measures and adhering to privacy guidelines, developers can protect user data from unauthorized access and breaches, while also fostering trust and reliability in the application.

2.11.1 Data Security

Encryption: Use encryption techniques to protect user data both at rest and in transit.

Authentication: Implement secure authentication methods like OAuth or JWT to verify users.

2.11.2 Privacy Concerns

- a. Compliance: Ensure compliance with data protection regulations such as GDPR and HIPAA.
- b. Privacy-Preserving Techniques: Implement techniques like data anonymization and differential privacy to protect user data.

2.12 Review of Similar Works

Jenkins et al. (1988) investigated the postprandial glycemia induced by different starchy foods and attributed these differences mainly to their rates of digestion. Low Glycemic indices (GI) food generate flatter glycemia which could have positive involvement in blood glucose control specifically on diabetics. Low-GI starchy foods also included legumes, pasta (made from durum or whole wheat), barley and parboiled rice; low-MuFAGI breads were mainly higher-fiber types such as pumpernickel, which is a type of whole-grain bread. The glycemic index concept, introduced in early 1980s allows for standardizing the post prandial (after meal) physiological response to oral glucose load and is based on a reference food which approximate white bread. This consistency allows for accurate distinction and cross-comparison of foods in varying studies, which can inform improved dietary management options such as those used to manage diabetes or

hyperlipidemia. It then covers the glycemic responses in real life, variation among individuals and other factors that influence their diet — such as whether foods are processed or not (highly relevant for full-time scientists) what they ate with each food to actually cook it! Although the GI has been criticized due to its lack of account for inter-individual differences as well as mixed-meal responses, it is valuable in identifying quality starchy foods and guiding dietetics advice. Their research provided crucial insights into how foods with lower GIs, such as legumes and whole grains, can help manage blood glucose levels. However, the study primarily focused on foods commonly found in Western diets, which may not fully encompass the dietary patterns seen in regions like Nigeria, where staples such as yam, plantain, and cassava are prevalent. While Jenkins et al. highlighted the importance of GI, they did not address the specific glycemic impacts of these local staples. This project aims to bridge that gap by focusing on the GI of Nigerian foods, offering a culturally relevant tool that better assists diabetic patients in Nigeria in making informed dietary choices based on the foods they commonly consume.

Zhang et al. (2015) developed "Snap-n-Eat" which is a mobile application that automatically recognizes food items, and calculates their caloric/nutritional value solely through images taken by the user's smartphone. The new system overcomes the limitations of earlier methods, which either needed extra input from users or controlled conditions and were not suitable for use in everyday life. It pre-processes images to segment food regions hierarchically and extracts low-level features such as color and texture coupled with the HOG & SIFT descriptors. The extracted features are encoded as Fisher Vector to improve robustness and a linear SVM classifier is trained for food item recognition, obtaining an accuracy over 85% in the validation with a dataset of 2000 images from 15 distinct classes. The system-use portion control by estimating how much a certain food should weigh or be based on the pixel count and assigning caloric/nutrition levels. This is in

contrast to past systems, which would not perform automatically across different environments and required training the system from scratch by giving it data about individual users. Plans to extend the range of detectable food categories and make it even more personalized, by learning user habits over time as they tag foods eaten at specific places or events would truly advance mobile dietary monitoring technology. However, while "Snap-n-Eat" excels in general dietary monitoring, it does not specifically address the unique needs of diabetic patients or the intricacies of local cuisines, particularly in non-Western contexts. In contrast, this project aims to extend the concept by focusing on foods typical to Nigerian diets, providing personalized dietary recommendations tailored to the specific needs of diabetic patients. By leveraging AI, the project will offer a more culturally sensitive tool that better serves the target population in Nigeria.

Yanai and Kawano (2015) used Deep Convolutional Neural Networks (DCNNs) to recognize food images, with pre-training and fine-tuning methods being applied successfully to improve classification accuracy was studied. They pre-trained DCNNs with the large-scale ImageNet dataset containing more than 1000 categories of food-related anchors, by obtaining feature representations from robust to specific features on these networks and fine-tuning them for a few datasets specifically may be satisfied (UEK-FOOD100, UEC-FOOD256). Using the above method, this technique obtained top-1 accuracy of 78.77% and 67.57% on UEC-FOOD100 & UEC-FOOD256 datasets outperforming existing benchmarks which gives it a rating over other methods. Compared to Kagaya et al. Compared to Yanai & Kawano (2014) which trained DCNNs from scratch, this method resulted in large improvement with pre-training and fine-tuning. In another open challenge, their study also revealed the scalability and effectiveness in large-dataset processing of DCNNs showing that they can be used for real-time tasks such as Twitter food photo mining. In this work, it was elucidated that transfer learning plays a major role in the food image

recognition task and pre-training on large scale dataset and fine-tuning with domain-specific data significantly improves performance. However, the datasets used in their study primarily consisted of foods common in Western diets, which limits the applicability of their findings to other cultural contexts. This project will build on their methodology by developing a specialized dataset that includes a wide variety of Nigerian foods, ensuring that the AI model is both accurate and relevant to the dietary habits of Nigerian diabetic patients. This approach will enable more precise food recognition and provide more appropriate dietary recommendations for this specific population.

Okamoto and Yanai (2016) Develop a novel calorie estimation system for smartphones that works solely on single image-based food recognition applications, so as to ensure correct functionality of the application even in offline state. For the system to effectively calculate food items/users much include a size-known reference item (like set of dice, shoe) in their meal photos. This model leverages the edge detection and k-means clustering for capturing initial food blob regions which are feed to accurate GrabCut algorithm in order segment exact region of foods. Based on the "Network in Network" architecture, it uses a Convolutional Neural Network (CNN) for classification and is able to run real-time while maintaining high accuracy using mobile devices. This contrasts with previous approaches like the one from Chen et al. in 2012 and Kong et al. Okamoto and Yanai did a similar work to ours on NutriNet, but instead of multiple images required for 3D volume reconstruction was utilized in DigitalOCTOPUS from 2012 using quadratic curve fitting to estimate calories [24], they only used pairs of weight-image relationships calculated by simple linear regression leaders in pharmacy dropout expenditures were significantly overestimated with the average absolute error at 52.231 kcal (21.3% relative error). The blend of accuracy and simplicity as offered by this system makes it a major leap for mobile food calories estimation technology. However, the need for a reference object may pose a usability challenge in

everyday situations, particularly in informal dining settings common in Nigeria. In contrast, this project seeks to enhance the user experience by developing an AI model that can estimate portion sizes without requiring additional objects in the image. By focusing on the unique serving methods and food presentation styles prevalent in Nigeria, the project will offer a more practical and user-friendly solution for diabetic patients in the region.

Shrimal et al. (2021) proposed a diet recommendation system designed to help users manage their food intake, reach weight-related goals, and generally improve their health. Their system uses Fuzzy Logic and Collaborative Filtering, two methods that allow for personalized food recommendations based on a user's dietary habits and preferences. One of the system's unique features is a Step Counter, which tracks physical activity and emphasizes the importance of regular exercise alongside a healthy diet. The process begins with users entering their BMI and diet preferences to set a calorie target. Then, the system continuously updates food rankings based on how frequently certain foods are consumed, adjusting recommendations accordingly. This method highlights the benefits of a structured diet and exercise routine, suggesting that personalized meal advice can have a significant impact on overall health. The system represents a promising tool for weight management and general health improvement, particularly through its use of Fuzzy Logic and Collaborative Filtering to customize food recommendations. The addition of the Step Counter reinforces the role of physical activity in maintaining good health. However, while the system effectively addresses general health and weight management, it doesn't specifically cater to the dietary needs of diabetic patients. In contrast, the project at hand goes a step further by incorporating AI algorithms specifically tailored for diabetes management, providing recommendations that account for both the glycemic impact and cultural relevance of foods common in Nigerian diets, offering a more specialized solution. While their approach is

comprehensive, it may not fully address the specific dietary needs of diabetic patients, particularly in the context of traditional diets like those in Nigeria. This project will build on their work by focusing more directly on the nutritional requirements of diabetic patients and incorporating localized dietary information relevant to Nigerian cuisine. By combining fuzzy logic with detailed nutritional analysis of Nigerian foods, the project aims to deliver more accurate and culturally appropriate dietary recommendations for diabetic patients.

Bhat and Ansari (2021) introduced an advanced machine learning framework aimed at predicting diabetes and offering personalized dietary recommendations for diabetic patients. This system leverages a variety of machine learning algorithms, including Decision Trees, Random Forests, and Naive Bayes, to analyze key health parameters like age, blood pressure, cholesterol levels, and hemoglobin. A significant aspect of their work is the emphasis on selecting the most relevant features to enhance model accuracy while reducing complexity. The data undergoes thorough preprocessing, ensuring that the predictive models perform optimally. The system integrates seamlessly with hospital management systems, utilizing patient data to generate personalized diet plans delivered via a mobile app. Metrics like precision, recall, F-measure, and accuracy indicate the models' effectiveness, with the Random Forest algorithm standing out for its reliability. The study also explores related works, highlighting the role of machine learning in healthcare. Future improvements are aimed at enhancing feature extraction and expanding the system's capabilities to manage additional health parameters and chronic conditions. While this study successfully demonstrates the integration of machine learning in healthcare, it tends to focus on general health parameters without delving deeply into the specific dietary habits of different regions. The proposed project enhances this approach by focusing specifically on dietary recommendations tailored to the Nigerian context, where traditional foods and local diets play a significant role in

diabetes management. However, their study predominantly focuses on general dietary advice and may not fully consider the cultural specificity required for different populations. In comparison, this project will offer a more targeted approach by specifically addressing the dietary habits and nutritional needs of diabetic patients in Nigeria. By integrating localized data and focusing on foods common in Nigerian diets, the project will provide more relevant and effective dietary guidance for this population.

Amugongo et al. (2022) conducted a thorough analysis of mobile applications that use computer vision for food detection, volume estimation, and caloric calculations—tools increasingly important for diet monitoring and managing chronic illnesses. Their research found that these apps are very accessible and utilize sophisticated AI technologies to help users accurately track their food intake. However, the study also uncovered some significant shortcomings. For instance, only one study attempted to explain the underlying features contributing to food classifications, and most apps (90.9%) failed to distinguish between food and non-food items, leading to a lack of transparency and user trust. The researchers categorized existing applications into different methodologies, noting that early mobile-based food recognition systems relied on traditional classifiers like Support Vector Machines (SVM) and K-Nearest Neighbor (KNN), which performed reliably. For example, in 2013, Kawano and Yanai achieved notable accuracy using SVM combined with manual features such as color and texture. Later, apps like "Snap-n-Eat," developed by Zhang et al. in 2015, achieved over 85% accuracy by utilizing a linear SVM classifier for hierarchical segmentation and feature extraction. More recent developments have focused on deep learning (DL), which offers better feature learning capabilities. Applications like Mezgec and Seljak's "NutriNet" and Park et al.'s Korean food classification model have demonstrated higher accuracy in food recognition tasks. Despite these advancements, issues such as image quality

variability and the need for multiple image views persist in 3D modeling for volume estimation. Additionally, the study highlighted the need for greater explainability in AI models to boost user confidence, pointing out that only a few studies employed methods like SHAP (Shapley Additive Explanations) to explain model predictions. Moving forward, the study suggests that future research should prioritize lightweight DL models for mobile use, integrate multiple data modalities, and enhance transparency and privacy protection. While these mobile applications show great promise, the study underscores the need for more inclusive datasets, standardized methods, and ethical guidelines to improve accuracy, reproducibility, and user trust. While they noted the strengths of such applications, they also identified a lack of transparency and challenges in distinguishing between food and non-food items. This project addresses these concerns by developing a more explainable AI model that clearly communicates how it generates dietary recommendations. Additionally, by focusing on the unique visual characteristics of Nigerian cuisine, the project will improve the accuracy and reliability of food recognition, enhancing user confidence and adoption in real-world settings.

Khan et al. (2022) developed a groundbreaking machine learning framework that predicts the glycemic index (GI) category of foods based on images—a vital tool for managing diets in individuals with metabolic disorders like diabetes. The researchers used the foodpics_extended database to classify foods into low, medium, and high GI categories. They extracted a variety of features from the images, including texture, statistical data, and shape, and evaluated the performance of five machine learning classifiers: AdaBoost with Random Forest (RF), J48 Decision Tree (DT), k-Nearest Neighbors (KNN), Naive Bayes, and Support Vector Machine (SVM) with Sequential Minimal Optimization (SMO). Their findings showed that the AdaBoost (RF) model delivered the highest overall accuracy. This innovative approach highlights the

importance of accurately estimating food size and volume using known reference objects within the images, which is essential for precise caloric and nutritional assessments. What sets this study apart is its unique focus on GI classification, whereas previous research primarily focused on estimating calorie and nutritional content. Khan et al.'s work lays the groundwork for future enhancements in dietary monitoring systems, demonstrating a robust methodology that can be refined with more diverse datasets to improve real-world applicability and accuracy. However, their research primarily used food images from a non-Nigerian context, which may limit the framework's applicability in regions like Nigeria. This project will build on their methodology by adapting the GI prediction to include Nigerian foods, ensuring that the system provides culturally relevant and accurate dietary advice for diabetic patients in Nigeria.

CloudDevs (2023) explored how machine learning (ML) can be integrated into React Native applications, which is particularly useful for developers looking to add sophisticated features like sentiment analysis and image recognition to mobile apps. The paper begins with the basics of setting up a React Native project and selecting the appropriate ML libraries, such as TensorFlow.js and TensorFlow Lite. Developers can train models using popular tools like TensorFlow or PyTorch, convert them into formats compatible with React Native, and integrate them for real-time image processing or predictive analytics. Key applications include sentiment analysis for user feedback and image recognition for tasks like analyzing food choices for diabetic care. The paper emphasizes the importance of best practices, such as asynchronous processing to keep the user interface responsive and implementing remote model updates for continuous improvement. While on-device processing is preferred for privacy and speed, it may be limited by the hardware capabilities of the device, whereas cloud processing offers more power but at the cost of latency and potential privacy concerns. The future of ML in React Native apps includes creating more

personalized user experiences, enabling real-time data processing, and enhancing security features. This integration empowers developers to create intelligent, adaptable mobile applications that evolve alongside user needs and technological advancements. While their discussion is technically sound, it focuses more on general applications rather than the specific needs of healthcare or dietary management. This project will apply their insights within the context of a mobile application designed for Nigerian diabetic patients, ensuring that the AI algorithms are optimized for both performance and relevance to the target audience. The project will also address any potential limitations in hardware capabilities, ensuring that the app runs efficiently on the types of smartphones commonly used in Nigeria.

Yera et al. (2023) conducted a systematic review of food recommender systems specifically designed for diabetic patients, categorizing various techniques and methodologies used in these systems. The review identifies four main categories of techniques: semantic-based, optimization-based, rule-based and categorization, and interaction-based systems. Semantic systems rely on ontologies and inference processes, while optimization approaches use methods like integer programming and metaheuristics. Rule-based systems and categorization approaches utilize IF-THEN rules and fuzzy logic, and interaction-based systems depend on real-time user data to make recommendations. The review highlights the strengths of these approaches, such as the use of well-defined frameworks and repeatable models, but also points out significant drawbacks like limited evaluation processes, the lack of comprehensive datasets, and the need for better integration of diabetes-specific knowledge. The authors suggest that future research should aim to create a unified framework that combines these different approaches, improves personalization by incorporating user preferences, and employs fuzzy tools and explainable AI to enhance the effectiveness and user confidence in these systems. Overall, this review contributes valuable

insights toward developing more effective and personalized nutrition guidance systems for diabetic patients. While their review is comprehensive, it points out some limitations, such as the lack of comprehensive datasets and the need for better integration of diabetes-specific knowledge. This project aims to overcome these limitations by developing a food recommender system that is specifically tailored to the dietary habits of Nigerian diabetic patients. By incorporating a robust dataset of Nigerian foods and leveraging advanced AI techniques, the project will provide more personalized and effective dietary recommendations that are culturally appropriate and scientifically sound.

Joshi et al. (2024) presented a comprehensive examination of AI applications in food science and nutrition, highlighting the transformative potential of AI in improving health outcomes through better dietary advice and more efficient food systems. The study emphasizes the importance of AI technologies, such as machine learning and deep learning, in analyzing large datasets to uncover patterns and correlations between nutrition and health. The authors discuss various AI applications, including food recognition, dietary assessment, and personalized nutrition, demonstrating significant improvements in these areas due to AI integration. The study categorizes AI applications in nutrition into several domains, showcasing substantial improvements in dietary assessments and personalized nutrition. AI tools have been shown to provide accurate nutrient analysis and recommendations, aiding health professionals in making more informed decisions. Despite its comprehensive coverage, the study highlights several challenges, including the need for ethical frameworks to manage data privacy concerns and mitigate algorithmic biases. While this work offers a broad overview of AI in nutrition but lacks focus on culturally specific dietary challenges, particularly for diabetic patients in regions like Nigeria, which my project directly

addresses by developing an AI-powered tool tailored to analyze Nigerian meals and provide personalized dietary recommendations

Important advances in dietary management and AI-driven tools are noted in the literature reviews, with different studies contributing to glycemic index analysis, food image recognition, and personalized diet recommendations. However, such efforts often stop short of addressing particular cultural and dietary contexts necessary for successful diabetes management in regions like Nigeria. Such gaps will be filled with the creation of an AI-powered tool that is localized specifically to Nigerian dietary habits and offers more accurate and relevant recommendations for diabetes patients. Such a tool is developed on the localized data and developed in line with common Nigerian foods so that it improves outcomes for the management of diabetes, bringing once again the very critical importance of cultural specificity in health tools and offering a solution that really does meet the needs of the Nigerian population.

CHAPTER THREE

METHODOLOGY

3.1 Introduction

This chapter outlines the methodology employed to achieve the project's objectives. Each section corresponds to a specific objective, detailing the processes and methods used to develop, implement, and evaluate the AI-powered dietary recommendation software.

3.2 Design and Development of a User-Friendly Software Interface

The user interface design for the AI-powered dietary recommendation software is crafted to be intuitive, accessible, and engaging, ensuring that diabetic patients can easily navigate and utilize the application. This section outlines the process for designing the user-friendly software interface, detailing the tools and technologies used, the wireframing and prototyping stages, and the integration of user feedback through iterative design.

3.2.1 Software Development Lifecycle (SDLC)

The Agile methodology was chosen for its iterative and incremental approach, allowing for flexibility and continuous improvement throughout the development process. The Agile methodology facilitated continuous feedback and improvement, ensuring the development of an effective AI-powered dietary recommendation mobile application. This approach allowed us to adapt and refine the application based on real user needs and feedback.

3.2.2 Phases Involved in the Development Process

The development process for the AI-powered dietary recommendation software was divided into several phases, employing the Agile methodology. React Native for the mobile application, Python for the backend API, and AI algorithms were used. In the Planning phase, the project scope and

objectives were defined, items were prioritized based on importance and urgency. In the Design and Development phase, both designing of the user interface using Figma and developing the mobile application with React Native, AI algorithms for image recognition and nutritional analysis were last in the stack. For Testing, unit, integration, and system testing, validated AI accuracy, were performed. During the Review and Retrospective phase, the completed features were demonstrated, feedback gathered, and improvements discussed for the next sprint. Finally, in the Deployment phase, features were deployed to a staging environment and stable increments released to production.

3.2.3 System Architecture

The system architecture for the AI-powered dietary recommendation software is designed to ensure scalability, reliability, and efficiency. The architecture integrates various components, including the user interface, AI algorithms, a nutritional database, and a recommendation engine, all working together seamlessly to provide personalized dietary advice for diabetic patients.

Figure 3.1 below is the diagram illustrating the system architecture.

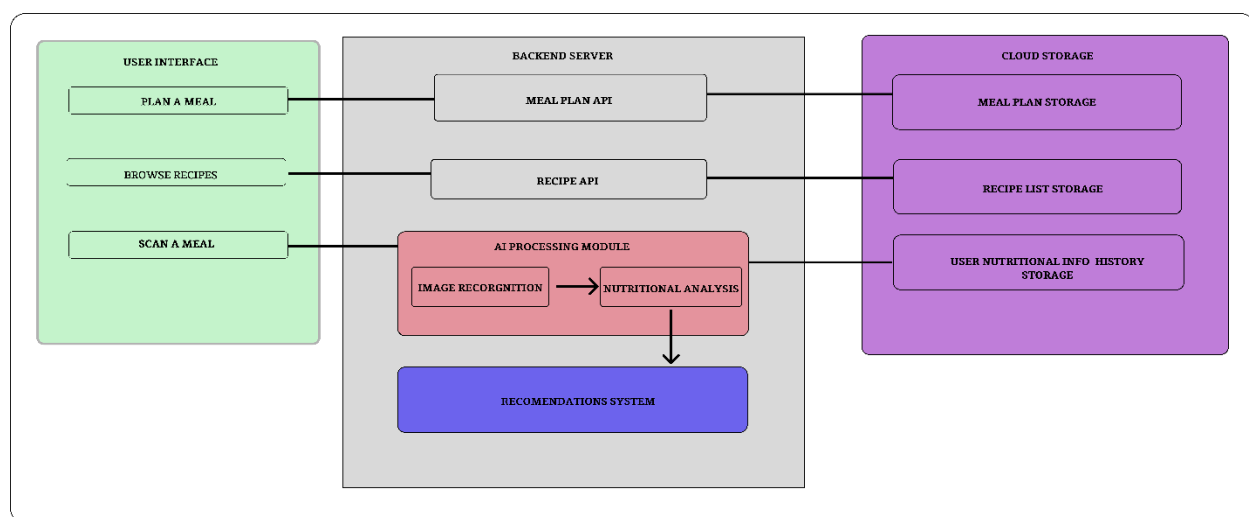


Figure 3. 1 System Architecture

3.2.4 Description of Each Component

Below is the detailed description of the various components of the system architecture

a. User Interface

The user interface is developed using React Native to ensure a smooth and responsive experience on both iOS and Android devices. It allows users to scan and upload images of their meals easily. It also has extra feature like common diabetic's friendly meal Recipes and diabetic health news feed to keep the user informed of the latest health updates on diabetics.

b. AI Processing Module

This module includes two main components:

- a. Image Recognition: Utilizes Convolutional Neural Networks (CNN) to identify and classify food items from the uploaded images.
- b. Nutritional Analysis: Employs Natural Language Processing (NLP) algorithms to analyze the nutritional content of the identified food items.

c. Nutritional Information Database

A comprehensive database that stores detailed nutritional information about various food items. This database supports the AI processing module by providing necessary data for accurate analysis.

d. Recommendation Engine

This is the core component that generates personalized dietary advice based on the

nutritional analysis and the specific dietary needs of diabetic patients. It uses advanced algorithms to tailor recommendations to individual users.

e. Cloud Storage

Ensures that all user data is stored securely in the cloud, with robust encryption and privacy measures in place to protect sensitive information. This component also manages data backup and recovery processes.

The system architecture is designed to provide a cohesive and efficient solution for analyzing meals and offering dietary recommendations, ensuring the software meets the needs of diabetic patients effectively.

3.2.5 Process for Designing the User-Friendly Software Interface

To create a seamless and interactive user interface, the following tools and technologies were used:

- a. Figma: Used for wireframing and prototyping, enabling collaborative design and real-time feedback.
- b. React Native: Employed for developing the mobile application, ensuring cross-platform compatibility and a responsive user experience.

3.2.6 Wireframing and Prototyping

The initial design phase involved creating wireframes and prototypes to visualize the layout and flow of the application. These wireframes served as blueprints, guiding the development process and ensuring all essential elements were included. Prototypes provided interactive simulations of the app, allowing for early testing and validation of design concepts. Figures 3.2, 3.3, 3.4, 3.5, 3.6, 3.7 show the meal plan screens, the authentication screens, home and account screen, scanning screens, scan history and scan result screens, recipe screens respectively.

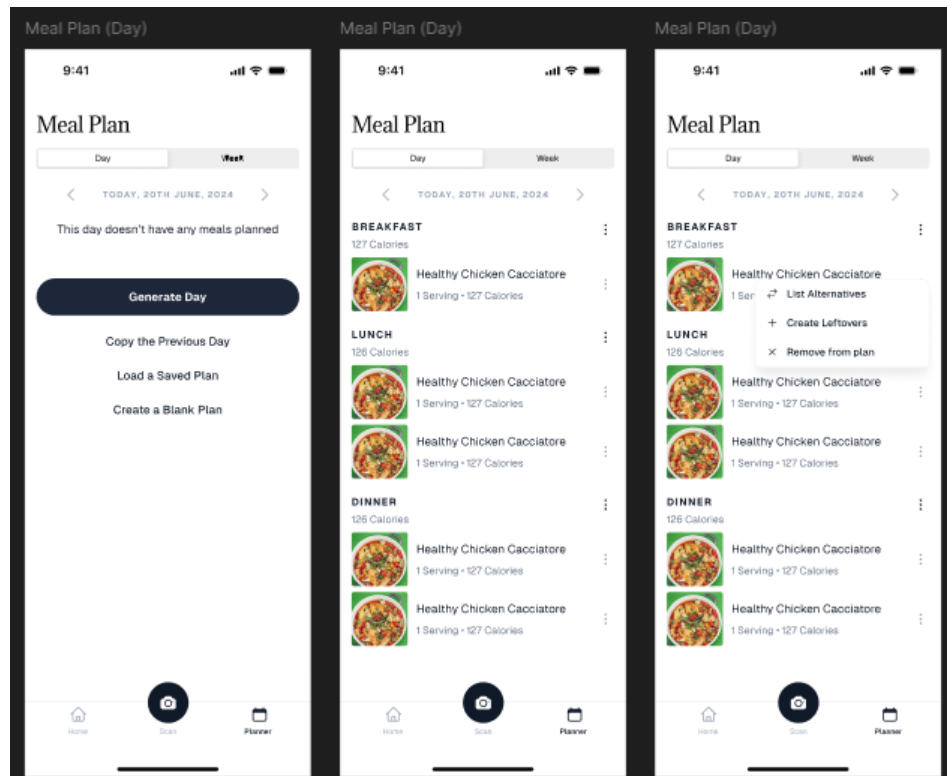


Figure 3. 2 Meal Plan Screens

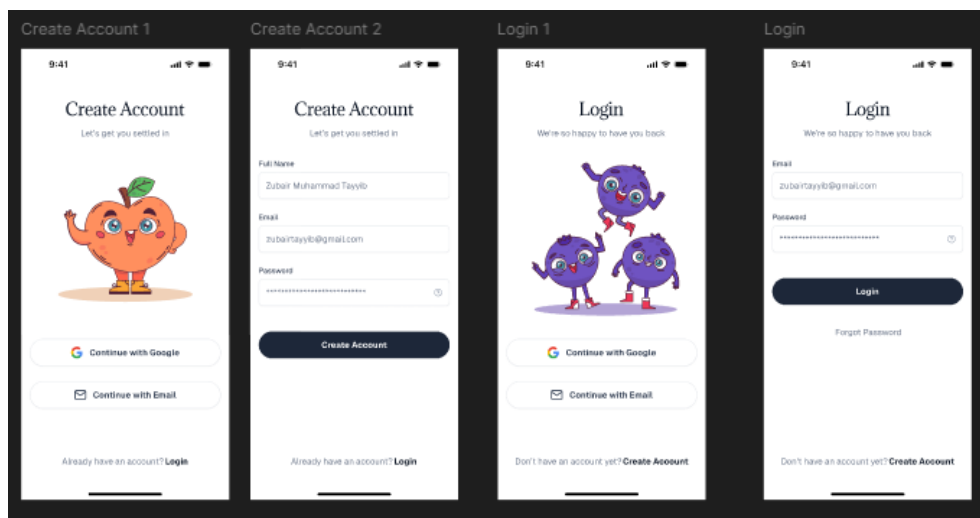


Figure 3. 3 Authentication Screens

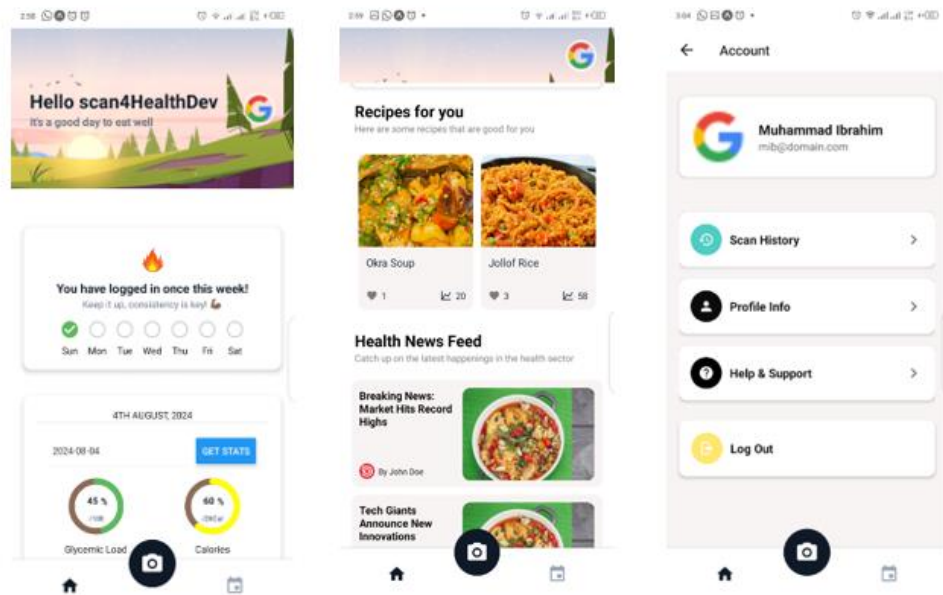


Figure 3. 4 Home Screens and Account Screen

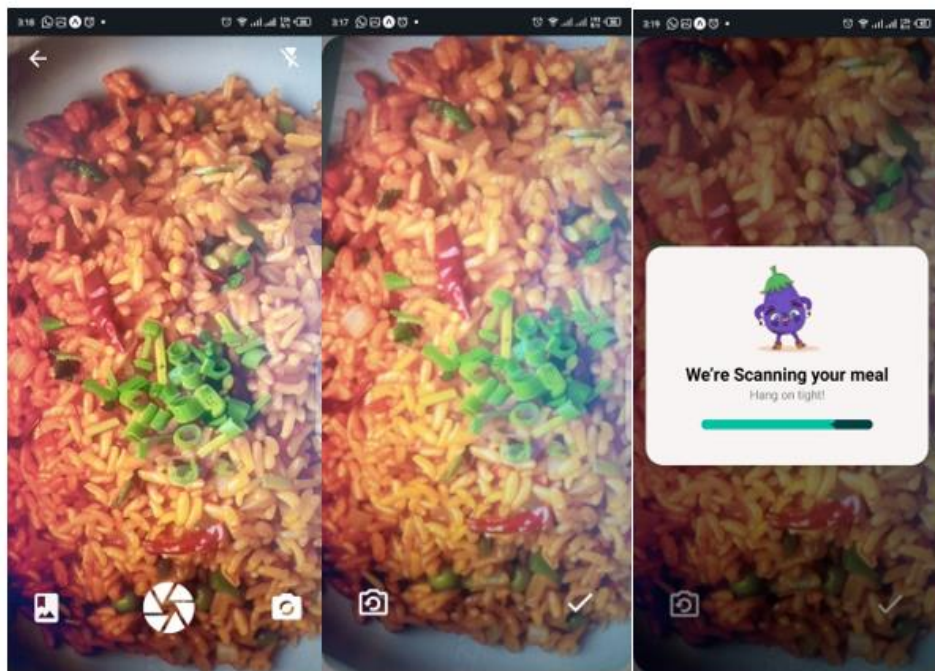


Figure 3. 5 Meal Scanning Screens

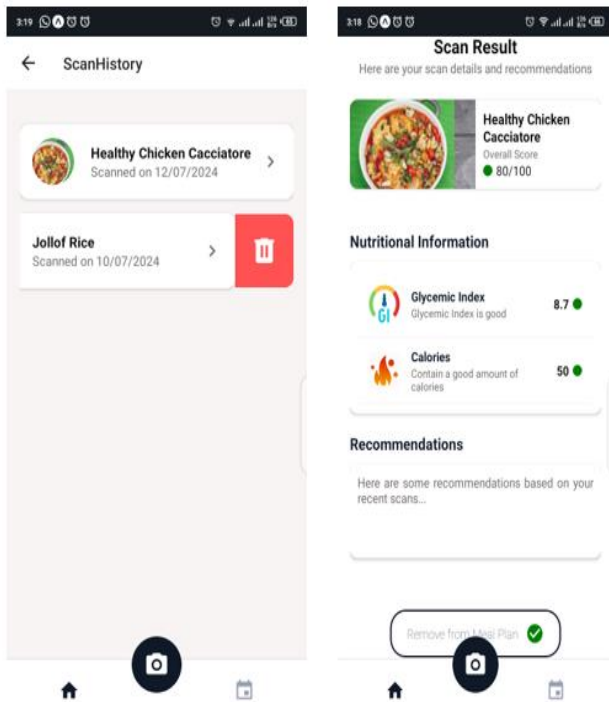


Figure 3. 7 Scan History Screen and Scan Result

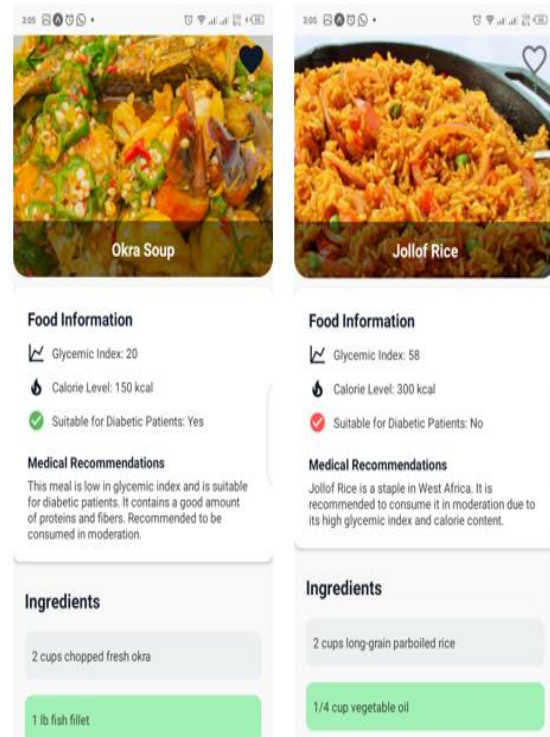


Figure 3. 6 Recipe Screens

3.2.7 User Feedback and Iterative Design

User feedback is pivotal to the iterative design process. After developing initial prototypes, conduction of usability testing sessions with a sample group of diabetic patients and healthcare professionals was carried out. This feedback informed several rounds of refinements, focusing on improving usability, accessibility, and overall user satisfaction. Below are different iterations of the meal plan Screen with figure 3.9 being the earlier design and figure 3.8 being the improved design upon user feedback.

Key steps in the iterative design process included:

- a. Usability Testing: Conducted sessions where users interacted with the prototype, providing valuable insights into user behavior and preferences.
- b. Feedback Analysis: Analyzed the feedback to identify common issues, pain points, and areas for improvement.
- c. Design Refinement: Made iterative changes based on the feedback, enhancing the user interface to better meet the needs of diabetic patients.
- d. Validation: Re-tested the refined prototypes to ensure that the changes addressed the identified issues effectively.
- e.

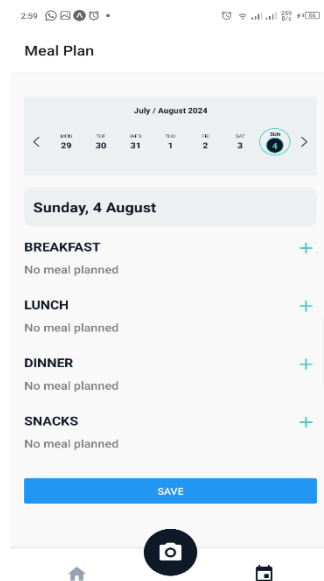


Figure 3. 8 Earlier Meal Plan Screen

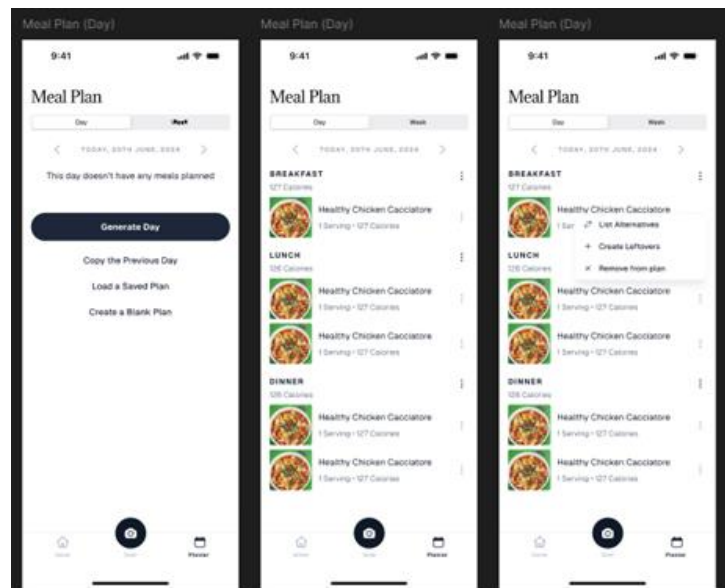


Figure 3. 9 Improved Meal Plan Screens

3.2.8 Mobile Frontend Development Using React Native

The mobile frontend development was a critical aspect of the project, ensuring that the application was accessible on both iOS and Android platforms. React Native was chosen for its ability to provide a consistent and performant user experience across multiple platforms.

3.2.9 Development Environment

Development environments used to build the mobile application are:

- a. Visual Studio Code (VS Code): The primary IDE used for coding, featuring extensions and tools that enhanced productivity and code quality.
- b. React Native Framework: Facilitated the creation of a mobile application with native performance using JavaScript and React.
- c. Expo: Was used for quick testing and deployment of the application during the development phase.
- d. GitHub: GitHub was used for code versioning.

3.2.10 Key Development Steps

The following are some key development steps taken to build the mobile application.

- a. Component-Based Architecture: Developed reusable components for different parts of the user interface, ensuring modularity and ease of maintenance. Below in Figure 3.10 is the components in the directory structure.

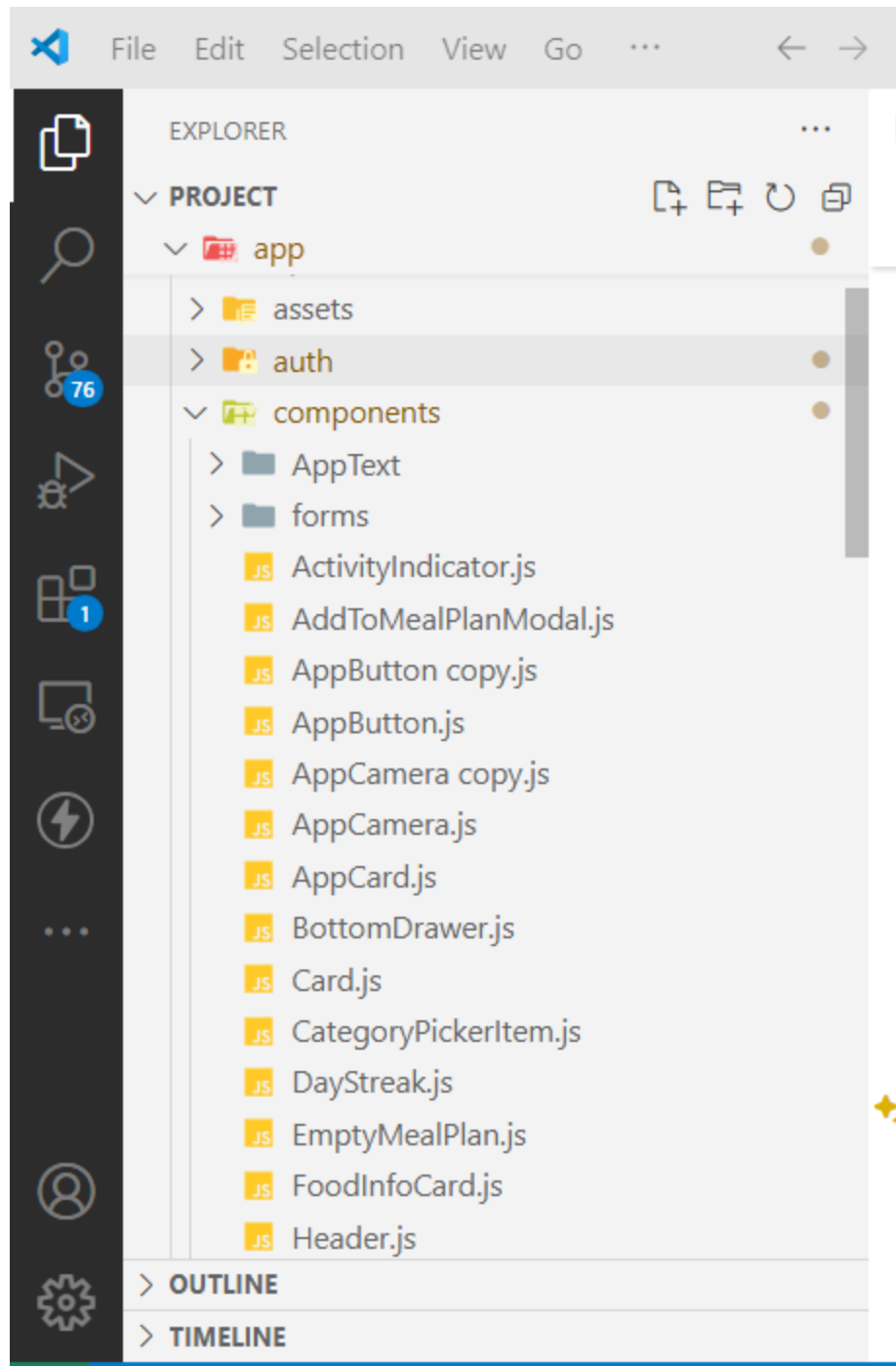
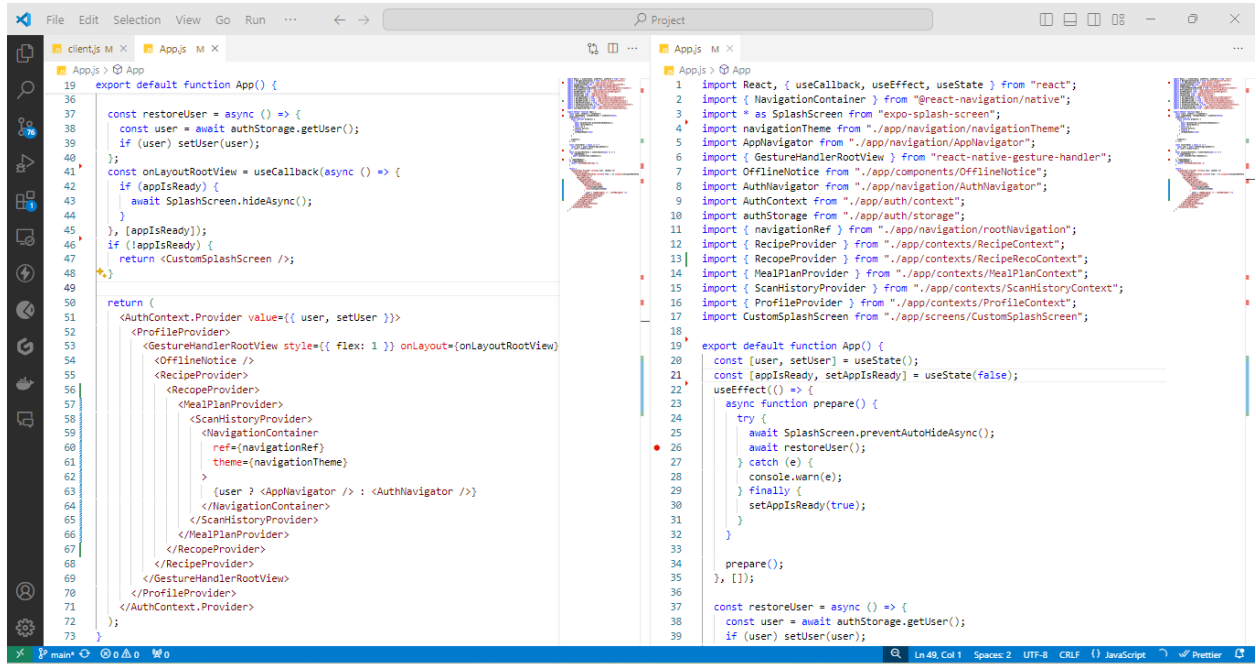


Figure 3. 10 Components in the directory structure

- b. State Management: Implemented state management using React hooks and context API to manage the application state efficiently. Shown below Figure 3.11 is the App.js with state managements.



```
19 export default function App() {
20   const [user, setUser] = useState();
21   const [appReady, setAppReady] = useState(false);
22   useEffect(() => {
23     async function prepare() {
24       try {
25         await SplashScreen.preventAutoHideAsync();
26         await restoreUser();
27       } catch (e) {
28         console.warn(e);
29       } finally {
30         setAppReady(true);
31       }
32     }
33     prepare();
34   }, []);
35   const restoreUser = async () => {
36     const user = await authStorage.getUser();
37     if (user) setUser(user);
38   };
39   const onLayoutRootView = useCallback(async () => {
40     if (appReady) {
41       await SplashScreen.hideAsync();
42     }
43     [appReady];
44     if (!appReady) {
45       return <CustomSplashScreen />;
46     }
47   }, [appReady]);
48   return (
49     <AuthContext.Provider value={{ user, setUser }}>
50       <ProfileProvider>
51         <GestureHandlerRootView style={{ flex: 1 }} onLayout={onLayoutRootView}>
52           <OfflineNotice />
53           <RecipeProvider>
54             <MealPlanProvider>
55               <ScanHistoryProvider>
56                 <NavigationContainer
57                   ref={navigationRef}
58                   theme={navigationTheme}
59                 >
60                   {user ? <AppNavigator /> : <AuthNavigator />}
61                 </NavigationContainer>
62               </ScanHistoryProvider>
63             </MealPlanProvider>
64           </RecipeProvider>
65         </GestureHandlerRootView>
66       </AuthContext.Provider>
67     );
68   };
69 }
```

Figure 3. 11 The App.js with state managements.

- c. API Integration: Integrated with backend services to fetch nutritional data and process user inputs seamlessly. The clien.js file where the frontend connects with the backend, is shown below in Figure 3.10.

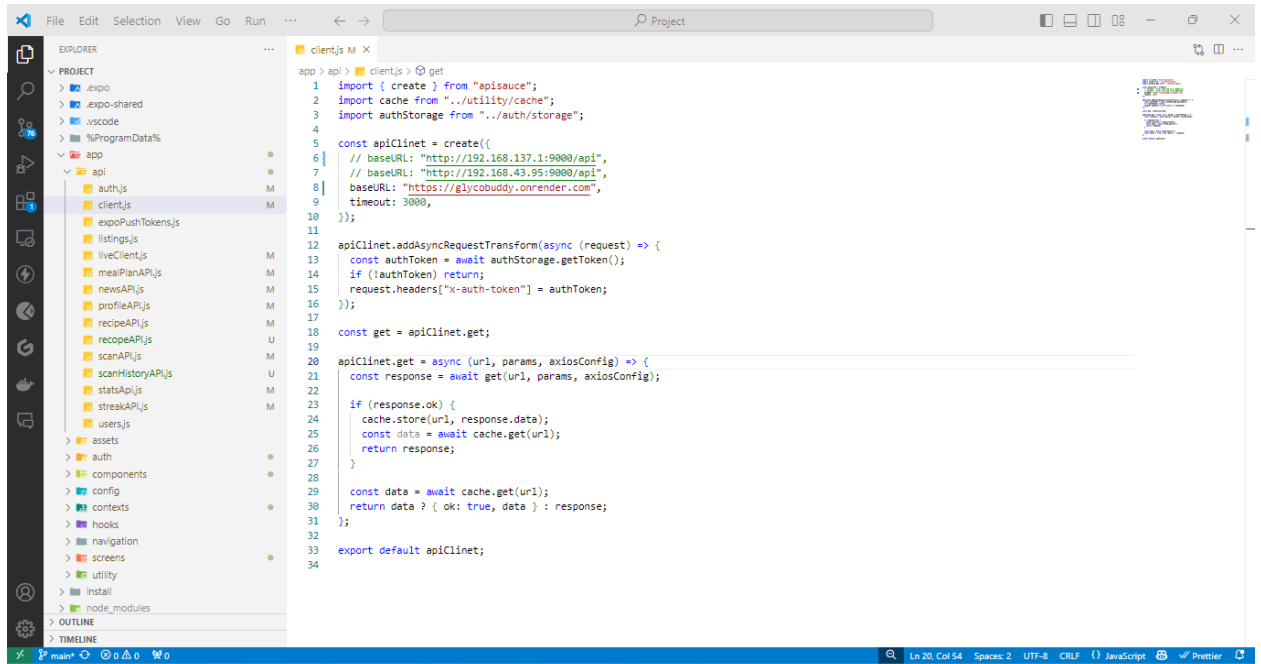


Figure 3. 12 Clen.js file where the frontend connects with the backend

- d. Responsive Design: Ensured that the application was responsive, providing a consistent experience on various device sizes and orientations.
- e. Testing and Debugging: Utilized tools like Expo's debugging tools and VS Code's integrated debugger to test and debug the application, ensuring a high-quality user experience.

Shown below (Figure 3.13 and Figure 3.14) are snapshots of the development environment.

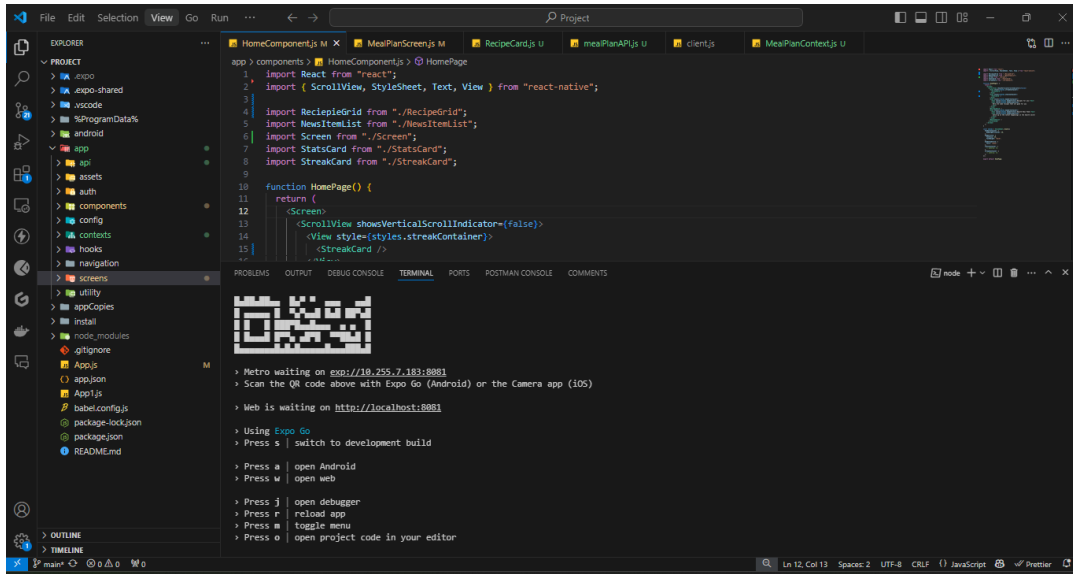


Figure 3. 13 Development environment showing VS Code editor starting an Expo React Native application

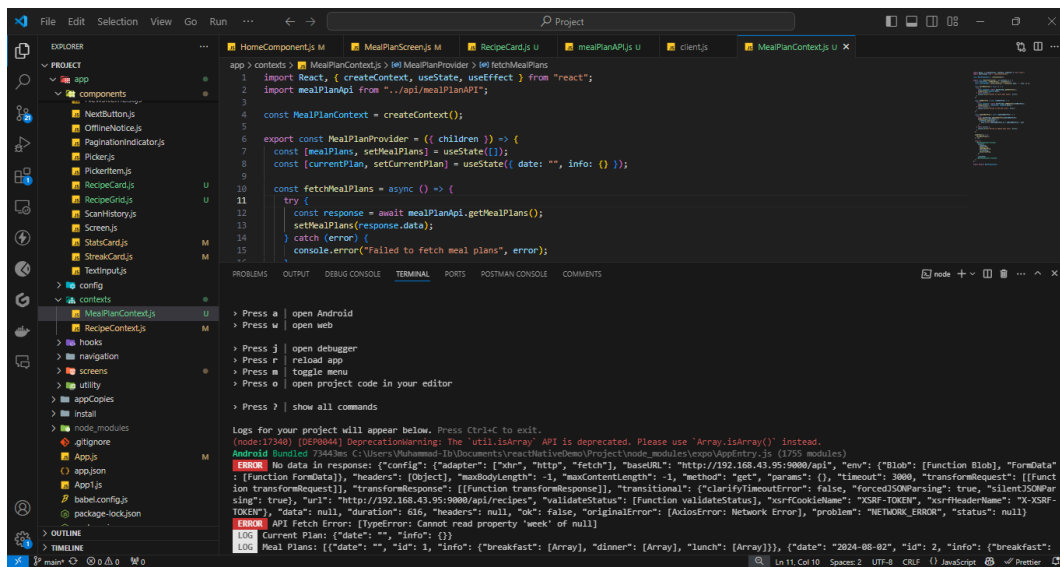


Figure 3. 14 Development environment showing VS Code editor running an Expo React Native application

The user interface design process for the AI-powered dietary recommendation software involved meticulous planning, prototyping, and iterative refinement. Using advanced tools like Figma, React Native, Git and VS Code, the development process focused on creating an intuitive and user-friendly application. By incorporating user feedback and adhering to best practices in mobile frontend development, the final product effectively meets the needs of diabetic patients.

3.3 Data Collection and Dataset Development

The data collection and database development phase of the dietary management application is a very important step and ensures the trained image classification model is accurate, and reliable. This section outlines the detailed process of data collection, including the acquisition and preprocessing of images for Nigerian dishes, as well as the development of the comprehensive dataset comprising 24 local Nigerian dishes with their corresponding Glycemic Index.

3.3.1 Data Collection Strategy

The first step in developing a comprehensive dataset involved gathering a wide array of images representing various Nigerian dishes. The goal was to create a diverse and representative dataset that covers the most commonly consumed foods in Nigeria, along with their corresponding nutritional information, such as glycemic index values. A list of 24 common Nigerian dishes from diverse regions and tribes of the country was collated. The list of foods considered includes Jollof Rice, Egusi Soup, Efo Riro, Banga Soup, Ofada Rice and Ofada Sauce, Edikang Ikong Soup, Amala and Gbegiri with Ewedu Soup, Ogbono Soup, Nkwobi, Afang Soup, Tuwo, Fried Plantain, Miyan Taushe, Beans Porridge and Plantain, Bitterleaf Soup, Ofe Nsala, Suya, Yam Porridge, Okra Soup, Eba, Moi Moi, Pepper Soup, Spaghetti, and Waina.

Web Scraping

A Python script utilizing Selenium and Chromium was developed to automate the process of web scraping from Pinterest and Google Images. These platforms were selected due to their rich repositories of food images, particularly of traditional Nigerian dishes. These codes are accessible from GitHub via <https://github.com/StarMindz/Pinterest-Image-Scraper> and <https://github.com/StarMindz/Google-Image-Scraper>

The script is designed for scraping both high-quality and low-quality images from the internet using a set of search terms. It includes codes for scraping full size images from Google. You can specify the number of images you wish to scrape and the directory on your computer where you'd like to store them. For each search term in your list, a folder will be automatically created with the same name as the search term, and the specified number of images will be downloaded automatically.

The scraped images were automatically categorized into folders named after the corresponding dish, ensuring that each food class is organized systematically. This step is important for the later stages of model training, where the CLIP model will learn to classify images based on these categories. Below in Figure 3.15 is the folder containing the project dataset.

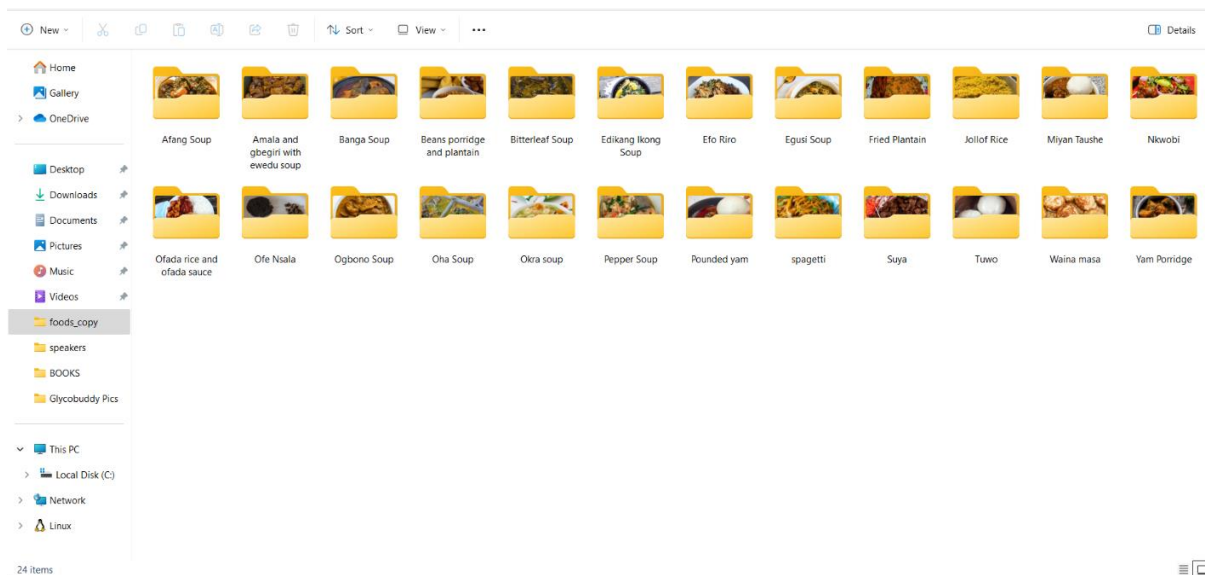


Figure 3. 15 Project Dataset

Data Cleaning

After collection, the dataset underwent a thorough cleaning process. Unrelated images, low-quality images, and irrelevant content were manually filtered out to ensure that only high-quality, representative images remained.

Glycemic Index Information

Each dish's glycemic index (GI) was researched and documented, primarily from reliable sources including scientific journals, nutritional databases, and studies focused on Nigerian cuisine. The GI data was then associated with each dish in the dataset. This dataset is used as part of the knowledge base for the Google Gemini LLM. Below in Figure 3.16 and figure 3.17 are the excel sheet of the collated meals and their respective parameters.

1	Food	Glycemic Index (GI)	Individual Glycemic Indexes	GI Classification
2	Afang Soup	40		Low
3	Amala and Ewedu Soup and Gbegiri	68.4	Amala (84.35), Ewedu/Jute leaves(20-25), Gbegiri(40)	High
4	Banga Soup	50		Medium
5	Beans Porridge and Plantain	55.2	Beans(40), Fried plantain(66-90)	Medium
6	Bitterleaf Soup	35.62		Low
7	Eba	82.25		High
8	Edikang Ikong Soup	23.64		Low
9	Efo Riro	14.05		Low
10	Egusi Soup	50-65		Low
11	Fried Plantain	69-90		High
12	Jollof Rice	98.9		High
13	Miyani Taushe	40		Low
14	Nigerian Pepper Soup	40-50		Low
15	Nkwobi	0		Low
16	Ofada Rice and Ofada Sauce	50		Low
17	Ofe Nsala	55-65		Medium
18	Ogbono Soup	40		Low
19	Oha Soup	40		Low
20	Okra Soup	17.02		Low
21	Pepper Soup	41		Low
22	Pounded Yam	81.60-82.6		High
23	Spaghetti	48 - 58		Low, Medium
24	Suya	0		Low
25	Tuwo	82 - 95.30		High
26	Waina Masa	70		High
27	Yam Porridge	65		Medium

Figure 3. 16 Collated meals and their respective arameters.

28	Abacha	84.88	High	Dried shredded cassava
29	Moi Moi	50.98	Low	Bean Pudding
30	Akara	44	Low	Bean Cakes
31				
32				
33				
34				
35				
36				
37	The glycaemic index (GI) ranks a carbohydrate containing food according to the amount by which it raises blood glucose levels after it is consumed in comparison with reference food (pure glucose or white bread),			
38	A GI of ≤55 is low, 56-69 is medium, and ≥70 is high, based on a glucose scale. ¹ The glycaemic load (GL) of a food is the GI multiplied by the available carbohydrate (g) in the serving divided by 100.2			
39	RESOURCES			
40	https://diabetesjournals.org/care/article/31/12/2281/24911/International-Tables-of-Glycemic-Index-and			
41	https://glycemic-index.net/glycemic-index-chart/			
42	https://www.bmj.com/content/374/bmj.n1651			
43	https://www.frontiersin.org/journals/nutrition/articles/10.3389/fnut.2021.687428/full			
44	https://guidelines.diabetes.ca/Guidelines/media/Docs/Patient%20Resources/glycemic-index-food-guide.pdf			
45	https://www.omicsonline.org/open-access/glycemic-index-of-selected-nigerian-foods-for-apparently-healthy-people-2165-7904_100016			
46	https://www.fitnessnigeria.com/glycemic-index-nigerian-foods/			
47				
48				

Figure 3. 17 Collated meals and their respective parameters (Continued).

3.4 AI Models Development

The AI models form the backbone of the system, enabling it to accurately recognize food items, assess their nutritional content, and provide personalized dietary recommendations for diabetic patients. The model development process involves the selection, and fine-tuning of state-of-the-art machine learning models to meet the specific needs of the application.

To achieve the desired level of precision and reliability, the development process was structured around two primary models: the Contrastive Language-Image Pre-training (CLIP) model, which handles image classification and food identification, and the Gemini Large Language Models (LLMs), which are responsible for processing natural language inputs and generating relevant dietary advice.

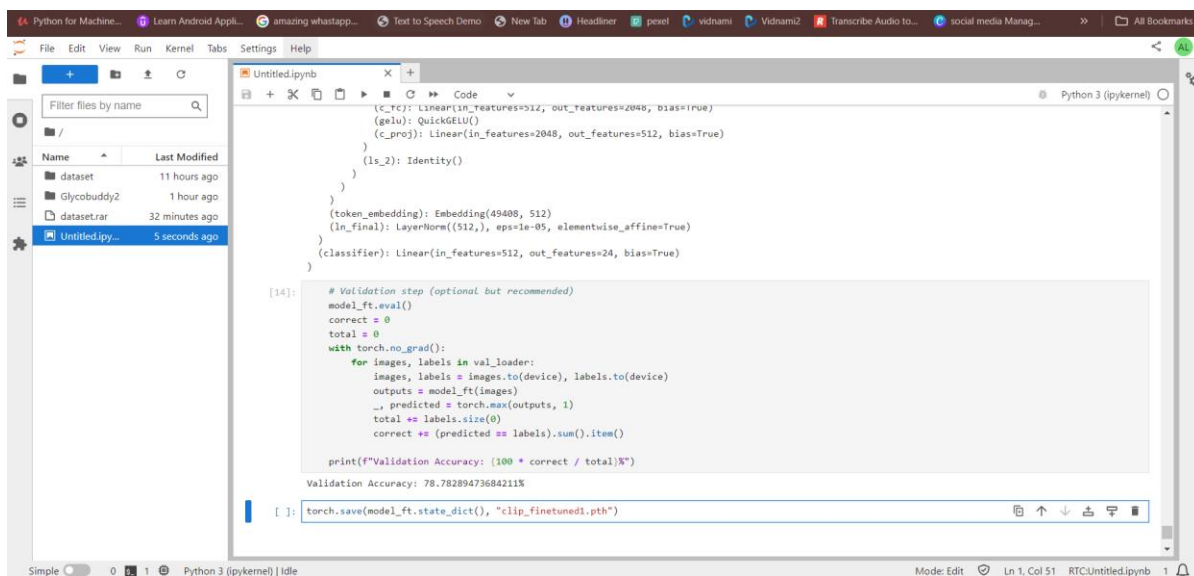
3.4.1 CLIP finetuning for image recognition

The fine-tuning of the CLIP model was conducted using the **open_clip_torch**, an open-source variant of the original CLIP model by OpenAI. This version provides more flexibility and is particularly well-suited for tasks that require extensive customization and fine-tuning, such as the specific requirements of this project. To handle the intensive computational demands of training a deep learning model like CLIP, the training process was executed on a cloud-based GPU environment provided by **Lambda Labs**. This environment was equipped with two NVIDIA A6000 GPUs, each with 48 GB of VRAM, alongside 24 CPU cores, 205.4 GB of RAM, and 1 TB of SSD storage. The use of GPUs was needed, as training the image recognition models on a CPU would have been very slow. The high-performance setup enabled the efficient processing of the large food image datasets and complex computations, accelerating the training process and making it feasible to fine-tune the model within a reasonable time frame.

The first step in the fine-tuning process involved carefully splitting the image dataset into training and validation sets to allow for proper evaluation of the model's performance. Data augmentation techniques such as random cropping, flipping, and color jitter were applied to the training data to increase variability and improve the model's robustness. These augmentations were designed to simulate the diverse conditions under which the model would need to perform in real-world scenarios, thereby enhancing its ability to generalize across different inputs.

Once the dataset was ready, the next step involved setting up the `open_clip_torch` model. The pre-trained CLIP model was loaded using the library, leveraging its initial training on a vast dataset of image-text pairs to provide a strong starting point. However, because the model was originally trained on a general dataset and not specifically on Nigerian dishes, the final classification layer needed to be adjusted to output predictions for 24 specific classes corresponding to the dishes in the dataset. The training process then began, utilizing a well-structured training loop that included multiple epochs, careful selection of batch size, and dynamic adjustment of the learning rate. The cross-entropy loss was chosen as the loss function, which is standard for multi-class classification problems. This loss function calculates the difference between the predicted class probabilities and the actual class labels, guiding the model's learning process. The Adam optimizer was used, which is known for its efficiency in handling large datasets and complex models. Adam adjusts the learning rate based on the first and second moments of the gradients, making it well-suited for deep learning tasks. The use of both A6000 GPUs in parallel allowed the model to process large batches of data efficiently, while the Adam optimizer ensured that the model converged to an optimal solution by adjusting the learning rate based on the gradients. After multiple training runs, adjusting the number of epochs, the different models generated were evaluated, and two

hyperparameters' conditions led to the highest performance, namely: training with an epoch of 10, which gave an accuracy of **78.78%** and training with an epoch of 20 which gave an accuracy of **84.37%**. For both cases, the batch size used for both training and validation was set to 32, and a learning rate of 0.0001 was used. Below in Figures 3.18, 3.19 are the model training screenshot using Jupita Notebook



The screenshot shows a Jupyter Notebook window with a file explorer on the left and a code editor on the right. The file explorer lists files: dataset (11 hours ago), Glycobuddy2 (1 hour ago), dataset.rar (32 minutes ago), and Untitled.ipynb (5 seconds ago). The code editor contains the following Python code:

```
(c_fc): Linear(in_features=512, out_features=2048, bias=True)
      (gelu): QuickGelu()
      (c_proj): Linear(in_features=2048, out_features=512, bias=True)
      (ls_2): Identity()
    )
  )
  (token_embedding): Embedding(40408, 512)
  (ln_final): LayerNorm([512,], eps=1e-05, elementwise_affine=True)
  (classifier): Linear(in_features=512, out_features=24, bias=True)
)

[14]: # Validation step (optional but recommended)
      model_ft.eval()
      correct = 0
      total = 0
      with torch.no_grad():
          for images, labels in val_loader:
              images, labels = images.to(device), labels.to(device)
              outputs = model_ft(images)
              _, predicted = torch.max(outputs, 1)
              total += labels.size(0)
              correct += (predicted == labels).sum().item()

      print(f"Validation Accuracy: {100 * correct / total}%")
      Validation Accuracy: 78.78289473684211%

[ ]: torch.save(model_ft.state_dict(), "clip_finetuned1.pth")
```

The output of the validation step is displayed below the code: "Validation Accuracy: 78.78289473684211%". The bottom status bar indicates the mode is "Edit", the cursor is at "Ln 1, Col 51", and the file is "RTC:Untitled.ipynb".

Figure 3. 18 Model training screenshot using Jupita Notebook (a)

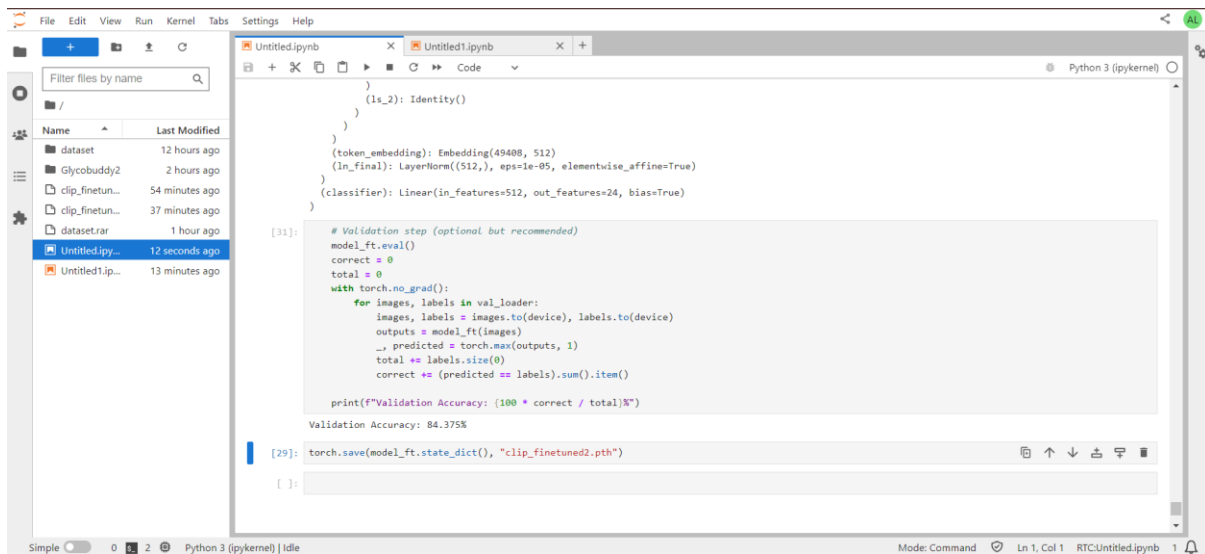


Figure 3. 19 Model training screenshot using Jupita Notebook (b)

In the final iteration of fine-tuning the CLIP model, a series of strategies that led to significant improvements were implemented. This technique includes the selective freezing of certain layers within the model, specifically, the lower layers of the model, which are responsible for capturing basic visual features such as edges, textures, and colors. The rationale behind this was to retain the feature extraction capabilities that the model had already learned during its pre-training on a large and diverse dataset. By freezing these layers, we ensured that the model did not overwrite these foundational features, which are need for better generalization and applicable across various image recognition tasks. Freezing these layers also allowed the training process to focus on fine-tuning the higher layers, which are more specialized and responsible for interpreting the basic features more specialized features needed in classifying the different local dishes. This approach not only sped up the training process but also increased the model accuracy to an impressive 99.84% while still retaining the model's generalization abilities.

3.4.2 CLIP Model for Image Classification

The Contrastive Language-Image Pre-training (CLIP) model, developed by OpenAI, is a multimodal learning model that connects text and images in a shared embedding space. CLIP is fine-tuned in this project to classify images of Nigerian dishes and associate them with their corresponding labels across 24 different classes. The goal is to enhance the model's accuracy in identifying these dishes, taking into account their unique visual characteristics.

- a. A curated dataset comprising approximately 10,000 images of 24 distinct Nigerian dishes is used for fine-tuning.
- b. Images are sourced through web scraping from platforms like Pinterest and Google Images, using a Python script that leverages **Chromium** and **Selenium** for browser automation.
- c. Scraped images are organized into folders named after their corresponding dish, ensuring each class is distinct.
- d. The dataset undergoes a rigorous cleaning process where low-quality or irrelevant images are manually removed to ensure the dataset's integrity.
- e. Preprocessing steps include resizing images to a uniform dimension suitable for model input and normalizing pixel values to enhance model training.
- f. Additional preprocessing techniques like data augmentation (e.g., rotation, flipping) may be employed to increase the dataset's variability and robustness during training.

- g. The CLIP model undergoes fine-tuning through several training epochs. During this process, the model's weights are adjusted to optimize its ability to distinguish between the 24 classes of Nigerian dishes.
- h. The fine-tuning process is conducted in a containerized environment using **Docker** to ensure consistency across different stages of development and deployment.
- i. A GPU from **Lambda Labs** is utilized to accelerate the training process, enabling the model to handle the computationally intensive task of learning from a large dataset.
- j. The model is fine-tuned and tested within this containerized setup, which not only facilitates easy deployment but also ensures the model can be scaled efficiently as needed.
- k. The Docker environment allows for reproducibility, making it easier to deploy the model across different platforms or environments without compatibility issues.

3.4.3 Gemini Large Language Models (LLMs)

The Gemini Large Language Models (LLMs), specifically Gemini 1.5 Pro and Gemini 1.5 Flash, are integrated into the application to handle natural language processing tasks. These models are responsible for providing accurate nutritional information of scanned food, providing meal recommendation taking into user profile, preferences and past meals, and interpreting user queries, engaging in conversational interactions through the application's chatbot. The Gemini LLMs are designed to process and understand complex language inputs, making them suitable for providing detailed and contextually relevant dietary advice. These models can handle a wide range of queries, from general nutritional information to specific questions about the glycemic impact of certain foods. One of the key challenges with LLMs is the risk of hallucination, where the model generates

inaccurate or misleading information. In this project, prompt engineering techniques are used to minimize hallucination, ensuring that the model's outputs are reliable and aligned with the user's dietary needs. To ensure that the Gemini models provide accurate and relevant information, significant effort is put into designing effective prompts. These prompts guide the model to generate responses that are not only correct but also useful for the user's context. The Gemini models are accessed via API calls from the backend. When a user interacts with the application, their input is processed by the backend, which then sends a query to the appropriate Gemini model. The model's response is then formatted and returned to the user through the front-end interface.

3.4.4 Google Gemini prompting for nutritional information and meal recommendations

The Gemini models played a pivotal role in providing detailed nutritional information and personalized recommendations for diabetic patients utilizing a knowledge base. The process of integrating these models into the application involved advanced prompt engineering techniques to achieve the desired functionality.

3.5 Development of the Project Back-End

The back-end is built using FastAPI, a high-performance web framework for building APIs. FastAPI is chosen for its ability to handle asynchronous operations efficiently, which is essential for the real-time functionalities of the application, such as food scanning.

3.5.1 API Layer

The APIs serve as the communication bridge between the front-end and the backend, facilitating operations such as user authentication, meal planning, food scanning, and interactions with AI models. FastAPI is integrated with Pydantic, which is used for data validation and serialization. Pydantic ensures that all incoming data conforms to the defined schemas, reducing the likelihood

of errors and enhancing the overall security of the application. This is especially important in managing user inputs related to dietary information, where accuracy is paramount.

3.5.2 Database Management

The application utilizes PostgreSQL as its primary database management system. PostgreSQL is chosen for its reliability, scalability, and support for complex queries, making it ideal for managing the diverse and structured data required by the application, such as user profiles, meal plans, nutritional data, and recipes. The database schema is carefully designed to ensure normalization and optimize query performance. Tables are structured to store information such as user profile data, meal plans, recipes, scan history, and interaction logs. The use of foreign keys and indexing enhances data integrity and query efficiency. To manage database schema changes, Alembic is used in conjunction with SQLAlchemy, the chosen Object-Relational Mapping (ORM) tool. Alembic facilitates database migrations by version-controlling schema changes, allowing for smooth updates and rollback capabilities. SQLAlchemy abstracts the complexities of SQL, enabling the development team to interact with the database using Python objects. This ORM approach enhances productivity and maintains consistency across different environments, while still allowing for raw SQL queries when necessary for performance optimization.

3.5.3 Security and Authentication

OAuth 2.0 is used for secure authentication, while role-based access control (RBAC) ensures that users can only access functionalities appropriate to their roles. JWTs (JSON Web Tokens) are used to securely manage user sessions, providing a stateless and scalable authentication mechanism. Passwords are hashed before being stored in the database and all data in transit between the front-end and backend is encrypted using SSL/TLS.

3.5.4 AI Models Integration

The AI models integrated into the dietary management application form the core of its functionality, enabling tasks such as food recognition, nutritional analysis, and personalized dietary recommendations. This section provides an in-depth exploration of the AI models employed, focusing on the CLIP model for image classification and the Gemini Large Language Models (LLMs) for natural language processing and dietary information retrieval.

3.6 Hosting and Cloud Infrastructure

The hosting and cloud infrastructure for the dietary management application is designed to ensure high availability, scalability, and reliability while leveraging the strengths of different platforms for specific components of the system. This section outlines the rationale behind the chosen hosting services, detailing the deployment of the backend, database, image storage, and AI integrations.

5.6.1 Render for Backend and Database Hosting

Render was selected to host the FastAPI backend and PostgreSQL database due to its simplicity, ease of use, and powerful features that streamline the deployment process. Render provides a straightforward platform for deploying web applications, with built-in support for various frameworks, including FastAPI. The deployment process is simplified, allowing for continuous integration and continuous deployment (CI/CD) workflows, ensuring that updates to the backend can be rolled out quickly and efficiently. Render also offers a managed PostgreSQL database service, which eliminates the need for manual configuration and maintenance of the database server. This service ensures that the database is always running smoothly, with automatic backups and updates, reducing the operational overhead.

3.6.2 AWS S3 for Image Storage and Processing

Amazon S3 (Simple Storage Service) is used for storing and processing images, including those uploaded by users for food recognition and classification. AWS S3 is known for its durability and scalability, making it an ideal choice for storing large volumes of food images. S3 integrates well with other AWS services, allowing for efficient image processing workflows. Images stored in S3 can be easily accessed by the CLIP model for classification, enabling real-time processing of user-uploaded images.

Below in Figure 3.20, showing the complete system infrastructure diagram.

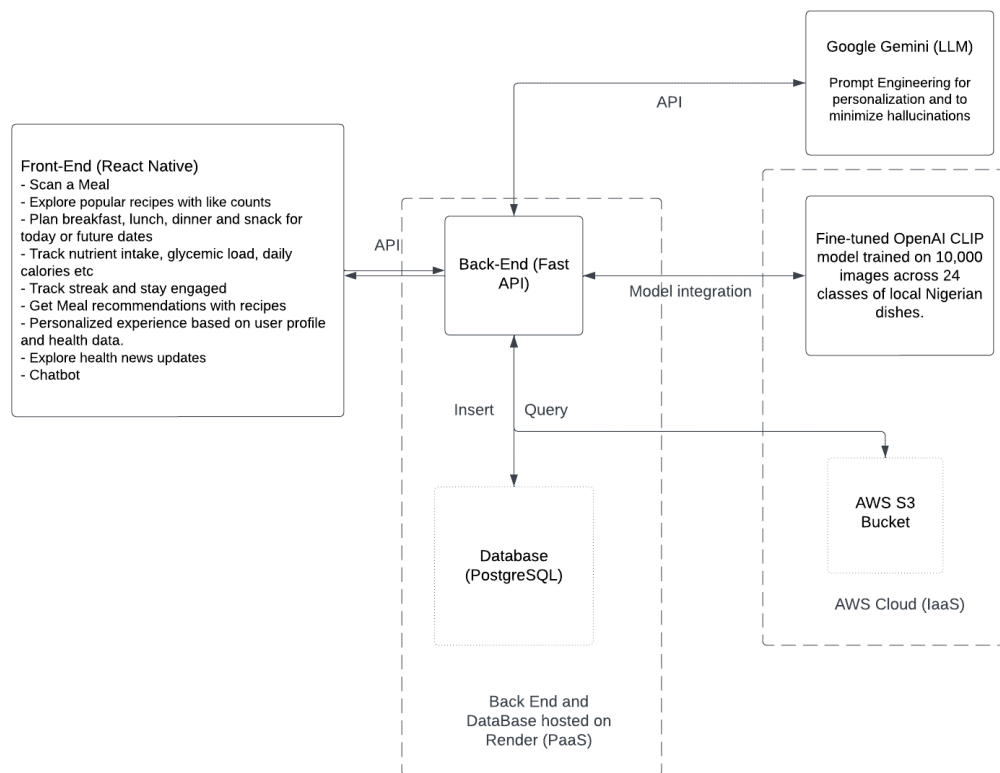


Figure 3. 20 Complete system infrastructure diagram.

This chapter presented the comprehensive methodology employed to achieve the project's objectives, focusing on each phase of the development process. It began with the design and development of a user-friendly software interface, highlighting the use of Agile methodology to ensure continuous improvement. The chapter also covers the systematic collection and preprocessing of data, the development and fine-tuning of AI models, the construction of a robust back-end system, and the strategic implementation of cloud infrastructure for hosting and storage. The methodology ensures that each component works cohesively to deliver accurate dietary recommendations for diabetic patients.