AI-Diet-Planner-App

Prof. D.V.Biradar[1], Prof K.A.Choure[2] Kasbe Krushna, Dnyanoba[3], Mane Om Dipak[4], Suryawanshi Vilas Balwan[5]

[1,2,3,4,5] *Dept of Information Technology, M. S. Bidve Engneering College, Latur, Maharashtra, India*

Abstract- **- Maintaining a healthy diet is essential for overall well-being, and tracking daily calorie intake plays a crucial role in achieving this goal. However, manually recording food items and estimating nutritional values can be time-consuming and inaccurate. The AI Diet Planner App is designed to automate the process of calorie and nutrition tracking, making it more efficient and user-friendly.**

The system uses Artificial Intelligence and Machine Learning techniques to analyze food images captured through a smartphone camera. By identifying food items from the images, the application estimates nutritional values such as calories, proteins, fats, and carbohydrates. The mobile application is developed using React Native**, while the backend and data storage are managed using cloud-based technologies. This automated approach reduces manual effort and helps users maintain a balanced and healthy diet effectively.**

Keywords- AI Diet Planner App, Mobile Application Development, React Native, Artificial Intelligence, Calorie Calculation, Nutrition Tracking, OpenRouter AI Model, Prompt-Based AI Processing, Convex Database, Cloud Backend, Real-Time Data Storage, Health and Fitness Application, Dietary Recommendation System, User Profile Management, Personalized Nutrition, Mobile User Interface Design, Cross-Platform Development, AI-Powered Health Monitoring

I. INTRODUCTION

With the rapid growth of mobile and artificial intelligence technologies, intelligent health applications play an important role in promoting a healthy lifestyle. Mobile applications enable users to track their daily activities and receive personalized guidance without the need for complex manual calculations. This project focuses on developing an AI Diet Planner App using React Native and cloud-based technologies. The system helps users calculate calorie intake, monitor nutrition, and understand diet planning concepts through AI-driven analysis and real-time data storage.

II. LITERATURE REVIEW

The AI-Diet-Planner-App has emerged as a significant tool in personalized nutrition management, leveraging artificial intelligence to provide customized diet plans based on individual health data, preferences, and lifestyle. Recent studies highlight the increasing use of AI in healthcare and nutrition, demonstrating its potential to improve dietary compliance, enhance health outcomes, and promote healthier eating habits.
AI-based diet planning applications enable functionalities such as calorie estimation, nutritional analysis, and personalized meal recommendations without requiring manual nutritional calculations [1], [3], [9]. Such systems are efficient to develop, test, and deploy, making them suitable for health monitoring, lifestyle management, and educational use in nutrition science and fitness domains [2], [6]. By integrating artificial intelligence models, these applications can dynamically adapt to user inputs and dietary goals, providing real-time insights and guidance.

III. PROBLEM STATEMENT

Traditional diet planning methods are complex and often require manual calorie calculations, nutritional chart references, and expert guidance. For beginners and individuals seeking a healthy lifestyle, there is a need for a simple AI-based diet planning system that provides accurate calorie and nutrition information without relying on manual record-keeping. Traditional dietary tracking methods depend heavily on handwritten logs or basic applications, which often lead to data inconsistency, inaccurate portion estimation, poor adherence, and increased chances of human error. As the number of people focusing on fitness, weight management, and health-conscious living continues to grow, managing daily food intake, calorie balance, and nutritional goals becomes increasingly complex and time-consuming.

Many existing diet planning solutions lack real-time AI-based analysis, centralized data management, personalized recommendations, and user-friendly interfaces, making it difficult for users to effectively monitor their diet and achieve health goals. Additionally, inadequate integration between food tracking, calorie calculation, and goal management modules can result in poor dietary decisions, limited nutritional awareness, and difficulty in tracking long-term progress and generating accurate reports.

Therefore, there is a need for an intelligent, reliable, and automated AI Diet Planner App that can calculate calories accurately, analyze nutritional intake, provide personalized diet recommendations, and help users maintain a healthy and balanced lifestyle using modern mobile and AI technologies.

IV. SYSTEM CONSTRUCTION

The AI Diet Planner App follows a simple and user-friendly workflow. The user enters food details manually or selects from predefined food options within the mobile application. These inputs are sent to the backend, where AI-based processing calculates calorie values and nutritional information.

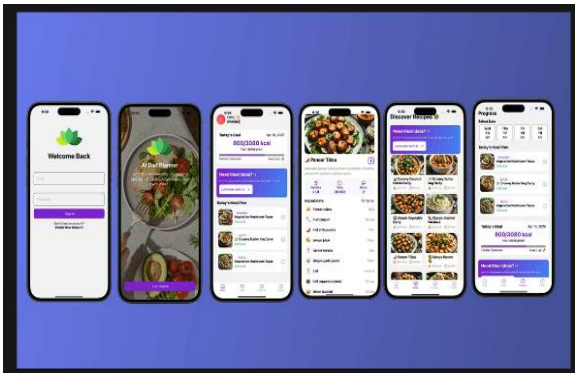

Fig 1-: System Construction

The user interacts with the application through a **React Native-based mobile interface**, where food details can be entered manually or selected from predefined food options available within the app. Inputs typically include food name, quantity or portion size, cooking method, and dietary goal. The user interface is designed to be intuitive, enabling easy data entry and minimizing user effort.

The AI-generated nutritional data is then parsed into a structured format and stored in the **Convex cloud database**. Convex provides real-time data synchronization, ensuring that calorie logs, meal history, and user progress are instantly updated and accessible across the application. This centralized storage supports daily summaries, progress tracking, and analytical features.

V. SYSTEM ARCHITECTURE

The architecture is divided into three logical layers:

1. Presentation Layer
2. Application Layer
3. Data Layer

Each layer performs a distinct function and communicates with adjacent layers to process user requests and generate accurate nutritional information.

A. Presentation Layer

The Presentation Layer is responsible for user interaction and interface rendering. It is developed using **React Native**, which enables cross-platform mobile application development for both Android and iOS devices.

Key functionalities include:

- Displaying user interfaces such as login, user profile, food entry, and calorie dashboard screens
- Capturing user inputs related to food details, portion size, and dietary goals
- Providing visual feedback and error messages

This layer ensures a user-friendly and responsive interface, enhancing usability and accessibility.

B. Application Layer

The Application Layer contains the core business logic of the system and is implemented using JavaScript. This layer processes user requests received from the presentation layer.

Major responsibilities include:

- User authentication and session handling
- Validation of input data
- Execution of banking operations such as balance inquiry, fund transfer, and loan application
- Managing system workflows and decision logic

The separation of logic from the user interface improves maintainability and reduces system complexity.

C. Data Layer

The Data Layer manages data storage and retrieval. For this implementation, client-side storage mechanisms such as local Storage or JSON-based data structures are used to simulate a database environment.

Stored data includes:

- Customer account details
- Transaction records
- User credentials

Although lightweight storage is used for demonstration purposes, the architecture supports integration with server-side databases such as MySQL or MongoDB in real-world deployments.

VI. METHODOLOGY

The first phase involves identifying the functional and non-functional requirements of the AI Diet Planner App. Functional requirements include user registration and authentication, food input and logging, calorie calculation, nutritional analysis, diet goal management, and progress tracking. Non-functional requirements focus on system usability, accuracy of AI-generated results, data consistency, performance, and security of user information. This phase ensures a clear understanding of user needs, dietary goals, and system constraints.

A. Requirement Analysis

The first phase involves identifying functional and non-functional requirements of the banking system. Functional requirements include user authentication, account management, fund transfer, loan application, and transaction history. Non-functional requirements focus on system usability, data consistency, security, and performance. This phase ensures a clear understanding of user expectations and system constraints.

B. System Design

Based on the analysed requirements, the system is designed using a three-tier architecture comprising presentation, application, and data layers. Logical models such as Data Flow Diagrams (DFDs) are used to represent data movement, while architectural diagrams illustrate component interaction. The design phase ensures separation of concerns and scalability.

C. Frontend Development

The user interface is developed using HTML and CSS to provide a structured, responsive, and user-friendly layout. Standard design principles are followed to ensure consistency across all modules. Forms are designed for secure data input, and navigation elements enable seamless user interaction.

D. Application Logic Implementation

Client-side logic is implemented using JavaScript, which handles input validation, business rules, and system workflows. JavaScript functions manage authentication, transaction processing, balance updates, and error handling. This approach enables real-time response without unnecessary page reloads.

E. Data Management

The system uses client-side storage mechanisms, such as local Storage and JSON-based objects, to simulate database functionality. User credentials, account details, and transaction records are stored and retrieved dynamically. This methodology supports fast data access and simplifies deployment in academic and prototype environments.

F. Security Measures

Basic security mechanisms are incorporated, including input validation, controlled access to system features, and session management techniques. Although client-side storage is used, the design supports future integration of encryption and server-side authentication for enhanced security.

VII. IMPLEMENTATION DETAILS

- Languages**:** React Native,
- Development Tool: Visual Studio Code, Android Studio Emulator, Convex
- Execution Environment: Android Studio Emulator

JavaScript variables and functions are used to handle transactions.

A. Programming Languages

The system is developed using HTML, CSS, and JavaScript, which together form the core technologies of the web-based implementation.

- HTML (Hypertext Markup Language) is used to define the structural components of the application, including forms for user login, account operations, fund transfers, and loan applications.
- CSS (Cascading Style Sheets) is employed to enhance the visual presentation of the system by providing consistent layouts, color schemes, and responsive design elements.
- JavaScript is used to implement the application logic, enabling dynamic interaction, input validation, transaction processing, and real-time updates without requiring page reloads.

B. Development Tool

The system is developed using Visual Studio Code (VS Code), a lightweight and powerful source code editor. VS Code provides features such as syntax highlighting, debugging support, code auto-completion, and extension support, which improve development efficiency and code quality.

C. Execution Environment

The application is executed in a standard web browser environment. Since the system is client-side, no additional server configuration is required. The browser interprets HTML for rendering the interface, applies CSS for styling, and executes JavaScript for handling business logic and data operations.

D. Transaction Handling Using JavaScript

JavaScript variables and functions are used extensively to manage banking operations. Variables store user account details, balances, and transaction data, while functions perform operations such as authentication, balance verification, fund transfer, and transaction history updates. Client-side storage mechanisms, such as local Storage, are used to persist data across sessions.

Conditional statements and event-driven programming techniques are applied to ensure accurate execution of transactions and proper error handling. This approach enables fast processing and immediate feedback to the user

VIII. RESULTS AND DISCUSSION

The system performs banking operations correctly on the frontend. Deposits and withdrawals update the balance instantly. Since no database is used, the

Following are the outputs as follow

Fig 3-: Login Page

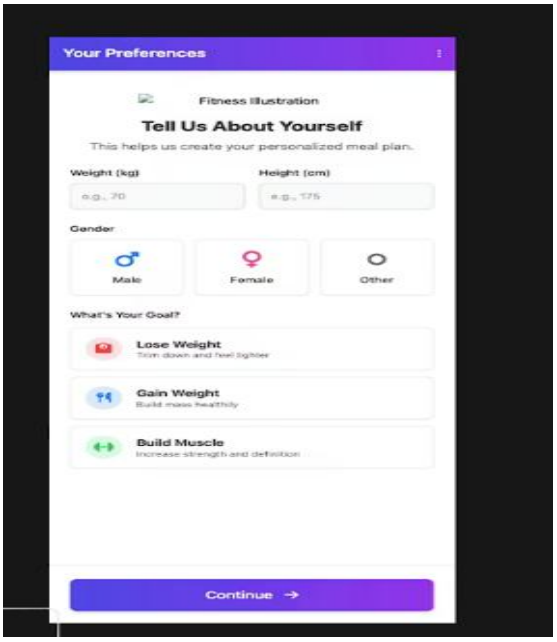

Fig 4-: Dashboard Page

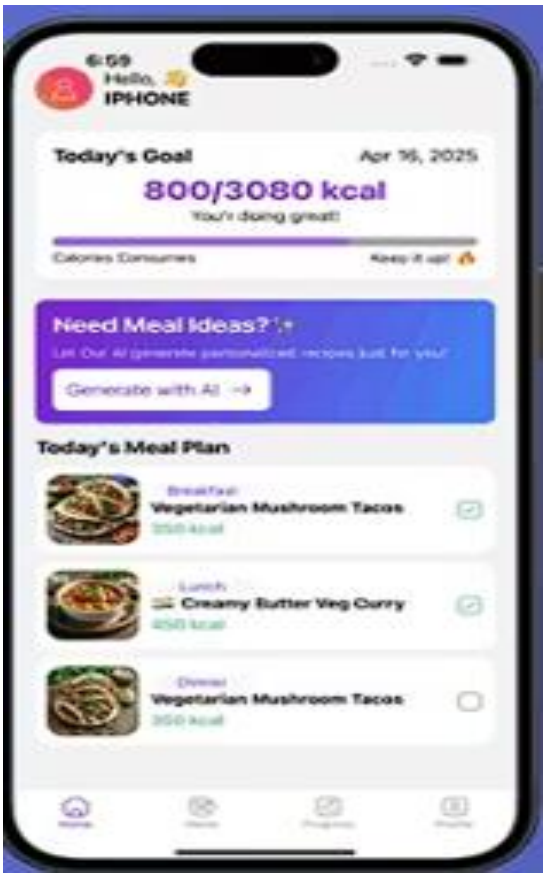

Fig 5-: AI Recipe Generate Page

IX. LIMITATIONS

A. Dependence on AI Model Accuracy

The system relies on an AI model to estimate calorie and nutritional values based on user-provided food details. As a result, the accuracy of outputs depends on the quality of user inputs, portion estimation, and food description. Variations in cooking methods, ingredients, and serving sizes may lead to approximate results rather than exact nutritional values. This limitation affects absolute precision in calorie tracking.

B. Lack of Authentication and Encryption

The proposed implementation does not include advanced authentication mechanisms or encryption techniques. User credentials and transaction data are handled without cryptographic protection, making the system vulnerable to unauthorized access and data breaches. This limitation prevents the system from meeting industry-level security standards.

C. Unsuitability for Real-World Banking Applications

Due to the absence of permanent storage, secure authentication, encryption, and regulatory compliance features, the system is not suitable for real-world banking operations. The application is intended for academic and prototype purposes only, where the primary focus is on understanding system architecture and functionality rather than meeting commercial banking requirements.

X. CONCLUSION

This paper presented the design and implementation of an **AI Diet Planner App** developed using React Native, cloud-based technologies, and an AI-powered calorie calculation model. The proposed system demonstrates how core diet planning functionalities such as food intake tracking, calorie estimation, nutritional analysis, and goal-based recommendations can be efficiently handled through a structured and modular architecture.

The adoption of a three-tier architectural approach improves system organization by separating the presentation layer, application logic, and data management components. This separation enhances maintainability, ease of development, and scalability of the system. The implementation highlights the effectiveness of artificial intelligence in providing dynamic nutritional insights and real-time calorie analysis within a mobile application environment.

This separation enhances maintainability, ease of development, and scalability.

The implementation highlights the effectiveness of client-side scripting in providing dynamic responses and real-time interaction within a browser environment.

Although the system successfully fulfils its intended academic objectives, certain limitations—such as lack of permanent data storage, absence of advanced security mechanisms, and limited real-world applicability have been identified.

These constraints indicate the necessity of server-side processing, secure databases, and encryption techniques for deployment in practical banking environments.

Overall, the proposed Bank Management System

serves as a functional prototype and an educational model for understanding web-based banking applications.

The system provides a foundation for future enhancements, including secure authentication, encrypted communication, and integration with backend services, making it a suitable base for further research and development.

XI. FUTURE SCOPE

A. Integration of Backend and Database

Future enhancements include integrating a robust server-side backend along with scalable cloud databases to improve data management and performance. Although the current system uses cloud-based storage, advanced backend services can enable complex analytics, enhanced data processing, and efficient handling of large user bases. This integration will support improved synchronization, backup, and long-term data management.

B. User Authentication

Advanced user authentication mechanisms can be implemented to ensure secure access to the system. This may include username–password authentication, role-based access control, multi-factor authentication, and session management techniques. Such mechanisms will prevent unauthorized access and enhance user trust.

C. Data Persistence

Implementing persistent storage through a backend database will ensure that user data and transaction records are permanently stored and retrievable across sessions and devices. Data persistence improves reliability, auditability, and long-term data integrity.

D. Improved Security Features

Security can be strengthened by incorporating encryption techniques for data storage and communication, such as hashing for passwords and secure communication protocols. Additional security measures may include input sanitization, access logging, and compliance with standard security practices.

E. Cloud Deployment

Deploying the system on a cloud platform would enhance scalability, availability, and fault tolerance. Cloud deployment enables remote access, load balancing, automated backups, and easier system maintenance, making the application more suitable for real-world banking scenarios.

REFERENCES

[1] Meta Platforms Inc., *React Native Documentation*. [Online]. Available: https://reactnative.dev/docs/getting-started

[2] OpenRouter, *OpenRouter AI Models and API Documentation*. [Online]. Available: https://openrouter.aiR. Pressman and B. Maxim, *Software Engineering: A Practitioner's Approach*, 8th ed., McGraw-Hill, 2015.

[3] Google Developers, *Mobile Application Development Fundamentals*. [Online]. Available: https://developer.android.com

[4] R. Pressman and B. Maxim, *Software Engineering: A Practitioner's Approach*, 8th ed., McGraw-Hill, 2015.

[5] I. Sommerville, *Software Engineering*, 10th ed., Pearson Education, 2016.

[6] Convex, *Convex Database Documentation* [Online]. Available: https://docs.convex.dev

[7] IEEE Computer Society, "Artificial Intelligence in Healthcare Applications," *IEEE Software*, vol. 37, no. 4, pp. 45–52, 2020.

[8] OWASP Foundation, *OWASP Top Ten Web Application Security Risks*. [Online]. Available: https://owasp.org/www-project-top-ten/

[9] U.S. Department of Agriculture, *FoodData Central*. [Online]. Available: https://fdc.nal.usda.gov

[10] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed., Pearson Education, 2021.

[11] OWASP Foundation, *Mobile Top 10 Security Risks*. [Online]. Available: https://owasp.org/www-project-mobile-top-10/