Smart Kitchen

Session: 2021 – 2025

Submitted by:

Qurat ul Ain 2021-CE-02

Noor Fatima 2021-CE-07

Ansa Aslam 2021-CE-21

Mohsin Qais 2021-CE-96

Supervised by:

Dr. Faisal Hayat

Department of Computer Engineering

University of Engineering and Technology

Lahore Pakistan

Smart Kitchen

(Session 2021 Computer Engineering)

Submitted to the faculty of the Computer Engineering Department of the University of Engineering and Technology, Lahore in partial fulfillment of the requirements for the Degree of

Internal Examiner
Signature:

Name:

Designation:

External Examiner
Signature:

Name:

Designation:

Chairman
Signature:

Dean
Signature:

Department of Computer Engineering

University of Engineering and Technology

Lahore Pakistan

Declaration

I declare that the work contained in this thesis is my own, except where explicitly stated otherwise. In addition this work has not been submitted to obtain another degree or professional qualification.

Signed: _____ Signed: _____

Date: _____ Date: _____

Signed: _____ Signed: _____

Date: _____ Date: _____

Acknowledgments

All acclamation and praises are for Allah Almighty, the Most Merciful and Compassionate for the strengths He gave us, and Holy Prophet Hazrat Muhammad (SAW), a model personality for the whole of mankind. Special thanks to our project supervisor, Prof. Dr. Faisal Hayat, for his invaluable advice, unwavering assistance, and remark- able understanding demonstrated throughout the project. His extensive knowledge and experience have been a constant source of inspiration for us. We would also like to extend our sincere appreciation to the faculty members of the Computer Engineering Department at the University of Engineering and Technology for providing us with the technical foundation and learning environment that enabled us to bring this project to life. We express our heartfelt gratitude to all individuals who have made direct or indirect contributions to our work. Your kindness and support mean a great deal to us. Lastly, we would like to thank ourselves and our parents for unwavering care, generous support, collaborative spirit, and continuous encouragement throughout the duration of this project.

Dedicated To our parents whose incredible support and prayers made us what we are today . . .

Contents

Acknowledgments iii

List of Figures x

List of Tables xi

Abstract xii

1 Introduction 1
- 1.1 Overview . . . 1
- 1.2 Background . . . 1
- 1.3 Motivation . . . 2
  - 1.3.1 The Challenge . . . 2
  - 1.3.2 The Potential of AI-Powered Smart Applications . . . 2
  - 1.3.3 Addressing Daily Living Challenges . . . 3
- 1.4 Objectives . . . 3
  - 1.4.1 Research Objectives . . . 3
  - 1.4.2 Industry Objective . . . 4
  - 1.4.3 Academic Objectives . . . 4
  - 1.4.4 Smart Objectives . . . 5
- 1.5 Problem Statement . . . 6
- 1.6 Scope of the Project . . . 7
  - 1.6.1 Socio-Economic Benefits . . . 9
    - 1.6.1.1 Enhanced Quality of Life: . . . 9
    - 1.6.1.2 Increased Accessibility: . . . 9
    - 1.6.1.3 Technological Innovation: . . . 9
    - 1.6.1.4 Flexibility: . . . 9
    - 1.6.1.5 Cost Efficiency: . . . 9
- 1.7 Challenges . . . 10
- 1.8 Assumptions and Constraints . . . 11
  - 1.8.1 Assumptions . . . 11
  - 1.8.2 Constraints . . . 11
- 1.9 Possible Applications of Work . . . 12
- 1.10 Beneficiaries of the Project . . . 12
  - 1.10.1 Primary Beneficiaries . . . 12

1.10.2 Secondary Beneficiaries . . . 13
1.11 Thesis Organization . . . 13

2 Literature Review **15**
2.1 Background of Smart Kitchen . . . 15
2.2 Evolution of Smart Kitchen . . . 15
2.2.1 Key Milestones in Smart Kitchen . . . 15
2.3 Benefits of Smart Kitchen . . . 16
2.3.1 Accessibility . . . 16
2.3.2 Flexibility . . . 16
2.3.3 Cost Efficiency . . . 17
2.3.4 Personalized User Experience . . . 17
2.4 Challenges in Smart Kitchen Implementation . . . 17
2.4.1 Technological Barriers . . . 17
2.4.2 User Interaction Challenges . . . 18
2.4.3 Social and Cultural Barriers . . . 18
2.5 Real-Time Monitoring and Feedback Systems . . . 18
2.5.1 Types of Monitoring and Feedback . . . 19
2.5.2 Technological Innovations in Assessment . . . 19
2.6 Smart Kitchen Literature Survey and Limitations . . . 19
2.6.1 Literature Survey . . . 20
2.6.1.1 Retrieval-Augmented Generation (RAG) in NLP Tasks (2020–Present) . . . 20
2.6.1.2 Kitchen Automation for Safety and Control (2022) 20
2.6.1.3 RAGAS Framework for Personalized Education 20
2.6.1.4 RAG-Based IT Support Chatbots (2024–Present) 20
2.6.1.5 PDF-to-AI Chatbot for Document Interaction (2024) 22
2.6.1.6 BIM Knowledge Enhancement with RAG for ChatGPT-4 (2024–Present) . . . 22
2.6.1.7 Automatic Exhaust Fan Systems in Kitchens (2024) 22
2.6.1.8 AI and ML in Sustainable Manufacturing (2000) 22
2.6.2 Integration of Smart Kitchen with Industry 4.0 . . . 22
2.6.2.1 Internet of Things (IoT) . . . 22
2.6.2.2 Artificial Intelligence (AI) and Machine Learning (ML) . . . 23
2.6.2.3 Big Data Analytics: . . . 23
2.6.2.4 Cloud Computing: . . . 23
2.6.3 Challenges in Implementing Industry 4.0 in Smart Kitchen . 23
2.7 Impact of AI-Powered Smart Applications on User Experience . . . 24
2.7.1 Key Findings . . . 24
2.8 Case Studies and Best Practices . . . 25
2.8.1 Case Study Examples . . . 25
2.9 Gaps in Existing Literature . . . 26
2.10 Conclusion . . . 26

3 Tools and Libraries **27**
3.1 Framework . . . 27
3.1.1 Flutter . . . 27
3.2 Backend Services and Libraries . . . 28
3.2.1 Firebase . . . 28
3.2.1.1 Realtime Database . . . 28
3.2.1.2 Data Storage and Sync . . . 29
3.2.1.3 Real-time Responsiveness . . . 29
3.3 AI Chatbot Module . . . 29
3.3.1 LangChain and RAG Implementation . . . 29
3.3.2 FAISS Vector Store . . . 30
3.3.3 OpenAI API and DuckDuckGo Search . . . 31
3.4 Embedded Hardware and Sensor Modules . . . 31
3.4.1 ESP32 Microcontroller . . . 31
3.4.2 Sensors . . . 31
3.5 Supporting Libraries and Packages . . . 32
3.5.1 Flutter Packages . . . 32
3.5.1.1 google_fonts . . . 32
3.5.1.2 provider . . . 32
3.5.1.3 transparent_image . . . 32
3.5.2 ESP32 Firmware Libraries . . . 32
3.5.2.1 WiFi.h . . . 32
3.5.2.2 FirebaseESP32.h . . . 33
3.5.2.3 DHT.h . . . 33
3.5.2.4 ArduinoJson . . . 33
3.5.3 Python Backend Libraries . . . 33
3.5.3.1 PyPDFLoader . . . 34
3.5.3.2 RecursiveCharacterTextSplitter . . . 34
3.5.3.3 dotenv . . . 34

4 Sustainable Development Goals (SDG's) **35**
4.1 Introduction . . . 35
4.2 Good Health and Well-being (SDG-3) . . . 35
4.3 Responsible Consumption and Production (SDG-12) . . . 36
4.4 Sustainable Cities and Communities (SDG-11) . . . 36
4.5 Industry, Innovation, and Infrastructure (SDG-9) . . . 37
4.6 Partnership for Goals (SDG-17) . . . 37
4.7 Conclusion . . . 37

5 Methodology **38**
5.1 Introduction . . . 38
5.2 Development Process . . . 38
5.3 Planning . . . 38
5.4 Design . . . 39

5.4.1 Database Design . . . 39
5.4.2 API Design . . . 39
5.5 Implementation . . . 40
5.6 Testing . . . 40
5.7 Deployment . . . 41
5.8 Architecture . . . 41
5.8.1 System Architecture Diagram . . . 41
5.9 Data Flow . . . 41
5.9.1 Data Flow Diagrams . . . 41
5.9.1.1 DFD Level 0 . . . 41
5.9.1.2 DFD Level 1 . . . 42
5.9.1.3 DFD Level 2 . . . 42
5.10 Use Cases . . . 42
5.10.1 Use Case Diagram . . . 42
5.11 User Flows . . . 44
5.11.1 General User Flow Diagram . . . 44
5.11.2 Chatbot Flow Diagram . . . 44
5.11.3 Sensor Flow Diagram . . . 44
5.12 Challenges and Solutions . . . 45
5.12.1 Technical Challenges . . . 45
5.12.2 Integration Challenges . . . 45
5.12.3 Performance Bottlenecks . . . 46
5.13 Conclusion . . . 46

6 Implementation **47**
6.1 Development Process . . . 47
6.1.1 Product Backlog Creation . . . 47
6.1.2 Sprint Planning . . . 47
6.1.3 Daily Stand-ups . . . 47
6.1.4 Sprint Execution . . . 48
6.1.5 Sprint Review . . . 48
6.1.6 Sprint Retrospective . . . 48
6.1.7 Repeat . . . 48
6.2 Development Technologies . . . 48
6.3 AI Chatbot with RAG-Based Architecture . . . 49
6.3.1 RAG Architecture . . . 49
6.3.2 Recipe Assistance . . . 50
6.3.3 Contextual Awareness . . . 51
6.4 IoT-Based Kitchen Monitoring System . . . 51
6.4.1 Temperature Monitoring . . . 51
6.4.2 Gas Detection . . . 51
6.4.3 Sound Detection (Baby Crying) . . . 52
6.4.4 Automatic Fan Control . . . 52
6.5 User Interface and Experience . . . 53

6.5.1 Recipe Catalog and Navigation 53
6.5.2 Favorites Screen 55
6.5.3 Filter Screen 56
6.5.4 Chatbot Screen 58
6.5.5 Real-Time Alert Notifications 60
6.6 Data Handling and Storage 62
6.6.1 Firebase Firestore 62
6.6.2 Firebase Realtime Database 62
6.6.3 FAISS Vector Database 63
6.6.4 Local Caching 63
6.7 System Testing and Validation 63
6.7.1 Unit Testing 64
6.7.2 Integration Testing 64
6.7.3 User Testing 64

7 Results and Conclusion 66
7.1 Introduction 66
7.2 Evaluation Overview 66
7.3 Technical Results 67
7.4 User Experience and Feedback 68
7.5 Future Enhancements 68
7.6 Conclusion 69

References 69

List of Figures

3.1 Flutter Framework 28
3.2 Firebase Working 28
3.3 Basic RAG 30
3.4 Working with FAISS 30
3.5 Working with ESP32 Microcontroller 31

5.1 Architecture Diagram 42
5.2 DFD Level 0 42
5.3 DFD Level 1 43
5.4 DFD Level 2 43
5.5 Use Case Diagram 43
5.6 Flow Diagram (User) 44
5.7 Flow Diagram (ChatBot) 44
5.8 Flow Diagram (Sensors) 45

6.1 Agile Development 48
6.2 RAG Architecture 50
6.3 IOT based Kitchen Monitoring 53
6.4 Categories Screen 54
6.5 Navigation Bar 55
6.6 Available Recipe List 56
6.7 Favorites Screen 57
6.8 Recipe Details Screen 58
6.9 Filters Screen 59
6.10 ChatBot Screen 60
6.11 Sensors Readings 61
6.12 Gas Leakage Notification 62
6.13 High Temperature Notification 63

List of Tables

2.1 Overview of Smart Kitchen Techniques and Limitations 21

Abstract

The project presents a smart mobile app that smoothly combines AI-based help for meals with live hardware monitoring of the kitchen. Built using Flutter and Dart, the app gives users a clever, customized place for finding meals, keeping recipes, and knowing about safety. The system is organized into four big parts: Meal Types, Liked Items, AI Helper, and Hardware Control. The main feature of the app is its AI-powered Chatbot, Made with a method called Retrieval-Augmented Generation (RAG), which is pretty advanced.

Unlike conventional static chatbots, the RAG-based system retrieves relevant meal-related information from a document knowledge base and combines it with the generative capabilities of large language models to produce contextually accurate, dynamic responses. If suitable information is not available locally, the chatbot automatically performs a fallback web search to ensure comprehensive assistance. Complementing the AI module is the Hardware Control feature, which provides optional, user-controlled real-time monitoring of the kitchen environment. Through the integration of temperature sensors, gas leakage detectors, and sound detection sensors connected via ESP32 microcontrollers and Firebase Realtime Database, the app offers live alerts to users. For example, if the kitchen temperature goes over a safe limit, the app turns on the exhaust fan and alerts the user. Likewise, gas leak detection or major sound activity activates immediate alerts in the app. Users are able to turn on or off each hardware component selectively based on their requirements. By integrating machine learning smarts with IoT-based real-time monitoring, this project provides a distinctive, user-focused solution that improves both kitchen safety and user convenience. It shows the power of integrating AI, cloud computing, and embedded systems as a single mobile platform, establishing a new benchmark for smart living applications.

Chapter 1

Introduction

1.1 Overview

With today's age of technological growth, the combination of Artificial Intelligence (AI) and the Internet of Things (IoT) has revolutionized mobile apps. The Smart Meals App with AI Chatbot and Real-Time Kitchen Monitoring consolidates intelligent meal suggestions and home safety monitoring into one application. Employing Machine Learning, namely Retrieval-Augmented Generation (RAG), the app includes a context-aware AI chatbot for meal-related support. In addition, it connects ESP32-based gas, temperature, and sound sensors through Firebase Realtime Database to detect the environment and provide real-time alerts. Through the amalgamation of AI-driven personalization and IoT-powered safety, the app enhances both convenience and home security through a single unified mobile experience.

1.2 Background

Smart home technologies and artificial intelligence (AI) have transformed the way people live in and interact with their living spaces, opening the door to increased convenience, safety, and personalization. Conventional home monitoring and kitchen management systems have generally been non-dynamic, providing minimal interaction and functioning autonomously without live integration into mobile applications. To address these shortcomings, the Smart Kitchen App with AI Chatbot and Real-Time Monitoring fills these gaps through an amalgamation of intelligent meal guidance and environmental protection in a unified platform. With a Machine Learning-driven chatbot based on Retrieval-Augmented Generation (RAG), users can obtain dynamic, context-based meal recommendations, while ESP32 microcontrollers connected to temperature, gas, and sound sensors

ensure continuous monitoring through Firebase. This system enhances both user interaction and household safety, representing a significant step toward personalized, intelligent living spaces.

1.3 Motivation

1.3.1 The Challenge

Despite the significant advancements in mobile applications and smart home technologies, users remain unable to access smart, real-time guidance that is tailored to their individual requirements. Conventional meal planning programs provide static, pre-formulated content with limited flexibility, while home safety monitoring devices often rely on standalone hardware and separate interfaces.

This fragmentation results in a disjointed and inefficient user experience. Moreover, with increasingly busy lifestyles, people seek solutions that not only help them make healthier meal choices but also ensure the safety of their living environments without requiring constant human oversight. The lack of integration between intelligent meal assistance and real-time kitchen monitoring highlights the urgent need for innovative solutions that can merge these essential services into a single, user-friendly platform.

1.3.2 The Potential of AI-Powered Smart Applications

Artificial Intelligence (AI) has made it possible to develop dynamic, context-sensitive applications. Techniques such as Retrieval-Augmented Generation (RAG) allow systems to intelligently retrieve relevant information and generate conversational, user-specific responses based on user queries. When applied to meal planning, RAG enables users to receive smart meal suggestions, personalized dietary advice, and ingredient-based tips through natural, human-like interactions. Simultaneously, the integration of IoT hardware components—such as temperature sensors, gas leak detectors, and sound detection modules—adds real-time environmental awareness to the system. By combining AI with IoT, modern mobile applications are now capable of delivering not only intelligent, context-aware assistance but also predictive safety features, effectively transforming the personal assistant into a truly smart companion for everyday life. This project leverages these emerging technologies to create a unified platform where users benefit from AI-driven meal guidance alongside optional, user-controlled real-time kitchen monitoring—ensuring both informed decision-making and enhanced household safety.

1.3.3 Addressing Daily Living Challenges

The Smart Meals App aims to solve several real-world challenges faced by modern users:

- **Intelligent Meal Planning:** Traditional apps require users to manually browse lists of meals. Our AI chatbot allows users to simply ask natural questions and receive immediate, personalized recommendations, drastically improving user convenience and engagement.
- **Proactive Safety Monitoring:** Kitchen risks like overheating or gas leaks can pose serious hazards. Incorporating hardware monitoring within the app ensures timely alerts in unsafe situations, eliminating the need for multiple systems or separate applications.
- **Unified User Experience:** By providing AI support and IoT command in one platform, users eliminate the hassle of fragmented systems, enjoying a seamless and streamlined digital experience.
- **Flexible User Control:** In acknowledgment of the importance of personalization, the application permits users to turn on or off individual hardware capabilities based on their personal choice, with complete mastery over the way the system accommodates their lifestyle.

  Through the integration of machine learning, real-time measurement, and mobile technology, this initiative serves to bridge necessary gaps in everyday digital living and provide a smarter, safer, and more personal user experience.

1.4 Objectives

1.4.1 Research Objectives

The main research goal of this project is to investigate the integration of state-of-the-art machine learning algorithms and IoT-based hardware monitoring into a single smart mobile application. This entails examining the potential of Retrieval-Augmented Generation (RAG) models in providing intelligent, document-driven meal support, alongside exploring the potential of embedded systems such as ESP32 in augmenting kitchen safety in real-time. Specific research objectives include:

- Investigating the effectiveness of RAG-based machine learning models in generating personalized, dynamic responses to meal-related user queries.
- Evaluating the impact of FAISS vector search and LangChain frameworks in improving document retrieval efficiency for chatbot functionality.
- Designing intuitive and accessible user interfaces that simplify user interaction with AI-powered services and hardware monitoring features.
- Assessing the role of real-time IoT communication (using ESP32 and Firebase) in improving user awareness and safety in kitchen environments.
- Exploring methods to ensure seamless synchronization between hardware sensors and mobile notifications for low-latency alert delivery.

1.4.2 Industry Objective

From an industry perspective, this project seeks to explore the market potential of combining AI-powered intelligent assistants with embedded real-time monitoring systems within mobile applications. As industries move toward smarter living solutions, understanding user demands for personalized interaction and home safety control is crucial. Key industry objectives include:

- Studying the adoption trends of AI-driven mobile applications and smart home safety monitoring solutions.
- Identifying technological innovations that drive the integration of AI, IoT, and mobile app ecosystems.
- Providing insights into the commercial viability of smart apps that combine personalized meal assistance with kitchen safety monitoring.
- Highlighting opportunities for expanding similar models into broader smart home, health, and wellness markets.

1.4.3 Academic Objectives

The academic objective of this project is to develop hands-on skills and theoretical understanding across multiple domains including machine learning, mobile application development, and embedded systems engineering. Specific academic goals are:

- To gain practical experience in implementing RAG-based AI models for real-world applications.
- To understand and apply Flutter and Dart for building cross-platform mobile apps with professional-grade UI/UX.
- To learn the design, integration, and management of embedded sensors (temperature, gas, sound) using ESP32 microcontrollers.
- To develop competency in real-time database management using Firebase Realtime Database.
- To foster professional responsibility, including timely project completion, risk management, and problem-solving under dynamic project conditions.
- To promote continuous improvement of system performance through iterative testing, optimization, and feedback-based refinement.

1.4.4 Smart Objectives

- **Specific:** Design and develop a mobile application that offers dynamic, AI-powered meal assistance alongside real-time user-controlled hardware monitoring features such as temperature, gas, and sound detection.
- **Measurable:** Achieve at least 90% success rate in the chatbot's ability to provide relevant, context-aware answers and ensure instant hardware alerts with notification latency of less than 5 seconds.
- **Achievable:** Utilize Flutter and Dart for rapid mobile development, integrate Firebase for reliable real-time data communication, and implement ESP32 hardware modules within project constraints.
- **Relevant:** Address the growing demand for intelligent, personalized mobile applications that also ensure user safety through proactive monitoring.
- **Time-bound:** Complete system development, testing, and deployment by May 2025, ensuring all modules are fully functional and tested.

The SMART objectives will guide the project's development, ensuring that deliverables are focused, measurable, achievable, aligned with real-world needs, and completed within the specified timeline.

1.5 Problem Statement

In today's more digitized world, the need for smart, customized mobile applications continues to grow. Several urgent issues have arisen, though, which highlight the imperative need for a smart, comprehensive solution that transcends the shortcomings of the conventional meal-planning apps and independent home monitoring systems.

Traditional meal suggestion apps mostly provide static recipe lists with minimal flexibility to accommodate user preferences or real-time personalized interaction. Such systems lack the ability to serve users who need dynamic, context-sensitive meal suggestions based on their nutritional requirements, available ingredients, or lifestyle options. Additionally, conventional chatbots employed in mobile apps are generally rule-based and do not have the intelligence to pull and create personalized, meaningful responses using current or varied sources of information.

Meanwhile, home security systems pertaining to kitchen settings (i.e., gas leakage detection or temperature sensing) are typically hardware dependent, breakable, and necessitate individual apps or devices for alerting and control. This fragmented experience leads to inefficiency, user inconvenience, and safety risks.

In addition, real-time response, reliability, and user control are essential in both intelligent information systems and safety monitoring uses. With the growing popularity of smart home technology, scalable, real-time cloud communications infrastructures are increasingly demanded to ensure smooth data exchange between hardware devices and mobile applications.

Identifying these challenges, the Smart Meals App development is intended to comprehensively solve the following issues:

The non-availability of dynamic, AI-based meal recommendation systems that can intelligently adapt to user queries.

The non-availability of context-aware chatbots using Retrieval-Augmented Generation (RAG) for delivering accurate and personalized meal support.

The fragmentation of kitchen monitoring systems, making users juggle several platforms for safety alerts and device control.

The requirement for cloud-based, real-time communication among IoT devices and mobile apps for immediate, accurate user notifications.

The necessity for user-defined hardware feature integration for flexible, personalized system behavior.

By integrating machine learning-based AI support with real-time, optional IoT monitoring, this project aims to provide an end-to-end smart living solution that improves user convenience, encourages healthier eating options, and maintains kitchen safety in one mobile app.

1.6 Scope of the Project

The project scope includes the design, development, and testing of a smart mobile app that combines AI-driven meal support with real-time hardware tracking for kitchen settings. The system is designed to provide dynamic, context-based meal suggestions utilizing the Retrieval-Augmented Generation (RAG) machine learning method while providing optional kitchen safety tracking capabilities such as temperature detection, gas leaks detection, and sound detection. The project aims at mobile users who are looking for intelligent food recommendation and real-time home security notifications, giving them a unified and personalized platform. It aims at utilizing up-to-date technologies like Flutter for mobile app development, ESP32 microcontrollers for IoT integration, and Firebase Realtime Database for cloud-based communication and notification delivery.

The Smart Meals App is conceived as an end-to-end solution providing:

- **AI-Powered Chatbot:** A smart conversational agent that dynamically loads and retrieves meal suggestions depending on user input using document-based retrieval and web search fallback.
- **Real-Time Kitchen Monitoring:** Optional, user-managed sensors for temperature, gas leak detection, and sound recognition, allowing for real-time alerts and automated functions such as switching ON/OFF the exhaust fan.
- **Meal Browsing and Favorites:** A structured meal directory that enables users to search recipes by categories (e.g., Breakfast, Lunch, Dinner) and mark favorite meals for fast future retrieval.
- **User Customization and Hardware Control:** Dynamic control panel for enabling/disabling individual hardware functionality based on individual user preferences.
- **Cloud-Based Data Synchronization:** Real-time synchronization between hardware the devices and the app using Firebase, providing instant notification and system reliability.

The project Smart Meals App is aimed at transforming the experience of users communicating with meal-planning systems and smart home safety devices and combining these two worlds into an intelligent mobile solution.

1.6.1 Socio-Economic Benefits

1.6.1.1 Enhanced Quality of Life:

By providing intelligent meal suggestions via AI and conducting real-time kitchen safety checks, the Smart Meals App greatly improves users' quality of daily life. Users are able to organize healthier meals in less time while reassured that any significant changes in their kitchen setting (e.g., overheating or gas leaks) will initiate instant alerts and auto-safety responses. Such empowerment promotes independence, efficiency, and safety especially for busy users, senior users, and families. **1.6.1.2 Increased Accessibility:**

The Smart Meals App is designed for broad accessibility: It operates entirely on mobile devices, requiring only a basic internet connection. It integrates with real-time cloud services (Firebase) to ensure fast, accessible communication. The system's modular hardware monitoring is optional, allowing users to adapt the app to their personal environment without needing expensive or complex smart home installations. Thus, the application is accessible not only in urban areas but also for users in semi-urban or less-connected regions who seek smarter daily assistance and home monitoring

1.6.1.3 Technological Innovation:

This project pioneers innovation by merging machine learning (specifically RAG for dynamic document-driven chatbots) with IoT sensor data streaming into a mobile ecosystem. It leverages cutting-edge technologies like: FAISS vector database search, LangChain document retrieval pipelines, ESP32-based real-time sensing, Flutter mobile frameworks, Cloud communication through Firebase. By demonstrating how AI, mobile development, and embedded systems can seamlessly work together, the project pushes the boundary of what intelligent apps can achieve in real-world living scenarios.

1.6.1.4 Flexibility:

The Smart Meals App is built with flexibility at its core: Users can enable or disable hardware features (temperature monitoring, gas sensing, sound detection) depending on their needs. Meal recommendations adapt based on real-time user queries, not fixed filters. The system design supports scalability and updates, allowing new sensors, documents, or features to be added easily in future versions. This flexibility ensures the app remains user-centric and adaptable to different lifestyles, environments, and future technology trends.

1.6.1.5 Cost Efficiency:

Compared to traditional smart home systems requiring expensive hubs or multiple apps, the Smart Meals App offers a cost-effective solution: Low-cost ESP32 modules and sensors minimize hardware expenses. Firebase's

scalable pricing ensures cloud communication remains affordable even as the number of users grows. By integrating AI meal assistance and safety monitoring into one mobile platform, the project eliminates the need for multiple subscription services or devices. The project demonstrates how intelligent solutions can be cost-efficient, scalable, and practical for a wide range of users.

1.7 Challenges

- **Access to Technology:** Ensuring that all users have access to compatible mobile devices and stable internet connectivity is crucial. The app relies on cloud services like Firebase for real-time notifications. Users in remote or low-connectivity areas may face difficulties in receiving timely updates. Compatibility across various Android and iOS devices must also be considered to provide a smooth user experience.

- **Digital Literacy and User Training:** Users need basic digital skills to interact with the app, such as enabling hardware features, chatting with the AI bot, and handling alerts. Although the app is designed to be simple and intuitive, some users may still require help. Offering in-app tutorials and easy-to-understand instructions will assist users with different technology familiarity levels

- **Engagement and Motivation:** Keeping users engaged with the AI chatbot requires delivering relevant and accurate responses. If the chatbot provides repetitive or incorrect answers, users may lose trust and interest. Regular updates to the RAG model and the knowledge base and a good fallback search system are required in order to keep users engaged.

- **Assessment and Integrity:** The dependability of real-time functionality such as temperature notification, gas sensing, and sound monitoring relies on sensor accuracy and quick data transmission. Delays or errors can impact user trust and safety. Thorough testing, optimizing, and effective error handling are paramount to ensure that the system operates well and precisely.

- **Technical Issues and Support:** Technical problems like server downtime, hardware malfunctions, or connectivity issues are inevitable in complicated systems. For users to remain satisfied, there should be timely technical support, frequent system upgrades, and convenient means through which users can report issues and get solutions instantly.

1.8 Assumptions and Constraints

1.8.1 Assumptions

The project takes for granted that users have fundamental digital literacy levels and have access to required technological tools, e.g., smartphones and reliable internet connection.

- **User Cooperation and Engagement:** The proper functioning of the Smart Meals App takes it for granted that users are inclined to engage actively with the application. Users will have to interact with the AI-facilitated chatbot for meal planning and activation of hardware features for real-time monitoring of the kitchen as required. Users are also expected to make use of the app's customization options in order to customize their experience.

- **Cloud Service Availability:** The application's architecture assumes that Firebase and other integrated cloud services will remain consistently operational and reliable. Any long-term disruptions in cloud services could affect real-time notifications and chatbot functionality, which would impact the overall user experience.

1.8.2 Constraints

Constraints are limitations that we must stick to when working on a project. They are the things that limit or restrict what we can do. This system has certain constraints that are:

- **Technical Limitations:** Device performance and network speed can vary significantly among users. Real-time responsiveness of notifications and chatbot interaction quality may be limited by lower-end devices, outdated operating systems, or unstable internet connections, affecting the user experience.

- **Population Access:**Accessibility to all users cannot be guaranteed, especially in remote or underserved regions where stable internet and modern smartphones are less prevalent. This geographical and technological gap could limit the widespread adoption of the application.

- **Data Quality:**The reliability of sensor data is a potential constraint. Sensor inaccuracies or calibration errors could result in false alerts or delayed notifications, affecting the trust and effectiveness of the real-time hardware monitoring system.

- **Operational Environment:** Ongoing internet connectivity and reliable power ply are required for the smooth operation of the app and the related hardware sensors. Where internet access or electricity is unreliable, the efficiency of real-time notifications and automatic hardware responses might be undermined.fications and automatic hardware actions could be compromised.

1.9 Possible Applications of Work

- **Individualized Meal Suggestion:** The Smart Meals App can suggest meals dividual user tastes, nutritional needs, and particular ingredient availability. Through the Retrieval-Augmented Generation (RAG) technique, the AI chatbot is able to suggest personalized recipes, meal plans, and healthy eating tips based on user searches. Dynamic personalization increases user interaction, encourages healthier eating, and results in a more fulfilling app experience.

- **Intelligent Kitchen Monitoring and Safety:**Hardware integration such tionalities like temperature sensors, gas leak sensors, and sound detection modules make it possible for the application to provide real-time kitchen safety monitoring. Users are able to get instant warnings of crucial environmental changes, which allows for timely preventive action. This preemptive safety function can avert accidents, thereby making kitchens safer, particularly for elderly users, children-families, and people juggling multiple tasks at once.

- **Real-Time Alerts and Environment Management:**Utilizing cloud-based timecommunicationusingFirebase,theappguaranteesusersreal-timealerts kitchenenvironment.Theautomatednatureofthesystemtotriggeractions,liketurningONanexhaustfanuponoverheating,provespotentialinreal-timeenvironmentmanagement.Thisfunctionalitycanfurtherbeextendedtoconnecttosmarterhomeautomationsystems,enablingaconnected,smartliving experience.

1.10 Beneficiaries of the Project

1.10.1 Primary Beneficiaries

The primary beneficiaries of this project are everyday mobile users, home cooks, and individuals seeking healthier lifestyle choices supported by intelligent technology. Users benefit from dynamic, personalized meal recommendations through the

AI-powered chatbot, helping them plan meals more efficiently and make healthier dietary decisions. Additionally, users who enable real-time kitchen monitoring features gain improved safety and convenience, receiving immediate alerts about environmental dangers such as overheat or gas leaks.

1.10.2 Secondary Beneficiaries

Secondary beneficiaries include developers of smart living applications, researchers in AI and IoT integration, and companies working on smart home solutions. The insights and methodologies developed through this project can guide future innovations in combining AI-powered personalized services with real-time hardware monitoring. The project's design approach also contributes to broader technological trends, promoting the creation of more intelligent, connected, and user-friendly mobile applications for smart living.

1.11 Thesis Organization

- **Chapter 1** Chapter 1 provides a general introduction to the Smart Meals App, including its objectives, motivation, background, challenges, and an overview of the problem it addresses.

- **Chapter 2** Chapter 2 is based on the research work carried out for the project, covering previous systems, related work, and the technologies that inspired and influenced the development of the application.

- **Chapter 3** Chapter 3 details the tools, libraries, and frameworks used to build the system. It covers technologies like Flutter, Firebase, ESP32, the FAISS vector search engine, and the LangChain framework used for implementing the Retrieval-Augmented Generation (RAG) chatbot.

- **Chapter 4** Chapter 4 discusses the Sustainable Development Goals (SDGs) linked to the project, particularly in terms of promoting health and well-being, technological innovation, and sustainable living practices.

- **Chapter 5** Chapter 5 explains the methodology adopted to design and develop the system, including the AI chatbot architecture, the hardware integration, the real-time data flow from sensors, and user interaction design.

- **Chapter 6** Chapter 6 is dedicated to the implementation, explaining how the suggested methodology was used in the real-world implementation and how each of the modules (Chatbot, Meal Categories, Hardware Monitoring) was created and tested

- **Chapter 7** Chapter 7 introduces the results, analysis, and conclusions from the project. It also addresses possible future improvements to the Smart Meals App, such as enhancing chatbot intelligence, hardware integration expansion, and system performance optimization.

Chapter 2

Literature Review

2.1 Background of Smart Kitchen

The convergence of artificial intelligence (AI), mobile development, and realtime monitoring has changed the way users engage with daily apps. AIpowered meal recommendation systems now provide personalized and context-aware suggestions through methods such as Retrieval-Augmented Generation (RAG). At at the same time, embedded platforms such as ESP32 and real-time databases like Firebase facilitate kitchen monitoring for improved safety. This chapter discusses the technologies of smart meal assistance, chatbot systems, and IoT-based real-time monitoring and emphasizes the necessity of an integrated solution such as the Smart Meals App that converges these technologies into one single platform.

2.2 Evolution of Smart Kitchen

The evolution of mobile technology, artificial intelligence (AI), and embedded systems has turned mobile apps into smart, interactive platforms. What used to be static functions like displaying recipes or simple notifications is no longer the case. With advancements in AI models, machine learning, and real-time cloud communication, their capabilities have grown exponentially.

2.2.1 Key Milestones in Smart Kitchen

- The launch of AI-powered chatbots enabled mobile apps to break past static responses to dynamic, context-sensitive interactions. Relying on rules, initial systems were shifted by the emergence of machine learning methods such as Retrieval-Augmented Generation (RAG) that allow applications to retrieve pertinent information and create tailored responses.

- Real-time cloud services like Firebase have made instant communication between mobile apps and embedded IoT devices. This has enabled users to observe real-world environments, such as kitchen safety conditions, from their phones directly.

- Mobile frameworks such as Flutter have also empowered developers to create cross-platform applications with rich, dynamic designs, while the inclusion of lightweight microcontrollers such as ESP32 has paved the way for embedded systems to integrate harmoniously with mobile platforms.

- In the future, the growing adoption of AI, Internet of Things (IoT), and real-time data analytics will further define the future of smart mobile apps, providing more intelligent, secure, and user-centric experiences.

2.3 Benefits of Smart Kitchen

Mobile applications based on AI that incorporate real-time monitoring have numerous advantages when compared to static apps. The advantages can be categorized under accessibility, flexibility, cost-effectiveness, and customized user experience.

2.3.1 Accessibility

- **Geographical Reach:** The Smart Meals App, which is powered by cloud services such as Firebase, enables users across various regions to utilize real-time meal suggestions and kitchen safety options. Even remote users can enjoy smart support and real-time notifications, as long as they have an internet connection.

- **Inclusivity:**The app has been kept as easy and intuitive as possible, so that available to users with varying levels of digital literacy. Simple toggle switches for hardware control and a natural language AI chatbot facilitate a broad spectrum of users, including those with less proficiency in sophisticated apps.

2.3.2 Flexibility

- **Self-Paced Learning:** Users are able to interact with the app at their own pace, questioning the AI chatbot whenever necessary or activating hardware features according to their choice. This flexibility integrates into the busy daily routines of busy lives, enabling them to schedule meal planning and kitchen monitoring without restrictions on time.

- **Anytime, Anywhere Access:** As the application is cloud-connected and based, customers can have access to services at any time and from anywhere. Whether at home or away, they get real-time notifications and customized meal suggestions, encouraging constant interaction

2.3.3 Cost Efficiency

- **Lower Costs:** Instead of installing high-cost standalone kitchen monitoring systems, users can employ basic ESP32-based sensors that are cost-effective yet efficient. Furthermore, utilizing Firebase and low-power hardware reduces maintenance and operational costs, thereby making smart safety solutions accessible to a large number of people.

- **Scalability:** The system is able to accommodate an increasing number of users without considerable infrastructure adjustments. New users may be included in the database, and more sensors can be accommodated with little effort, thus making the solution scalable for expansion.

2.3.4 Personalized User Experience

- **Adaptive Assistance:** With the employment of Retrieval-Augmented Generation (RAG) models, the AI chatbot is able to personalize responses according to the user's questions, likes, and surroundings. This makes the meal suggestion process dynamic, enhancing relevance and enjoyment.

- **Immediate Feedback:** Immediate notifications from sensors, along with real-time chatbot interactions, offer users real-time feedback on kitchen safety as well as meal suggestions. This real-time feedback loop ensures that users remain consistently informed, safe, and aided.

2.4 Challenges in Smart Kitchen Implementation

Although providing several advantages, AI-facilitated mobile apps with hardware integration also encounter a number of challenges that have to be managed in order to guarantee their effectiveness and dependability.

2.4.1 Technological Barriers

- **Digital Divide:** Smartphone and stable internet connections are necessary for effectively using the Smart Meals App. Underserved or rural users can

experience challenges when using real-time cloud services, restricting the accessibility and influence of the app.

- **Technical Issues:** Ensuring stable performance in both the AI chatbot and hardware monitoring systems is a technical challenge. Server failure, sensor errors, cyber attacks, and data privacy issues need to be addressed prudently to maintain user confidence and make the platform reliable.

2.4.2 User Interaction Challenges

- **Engagement:** Engaging users with the AI chatbot and other app features needs to be dynamic, relevant, and personalized. When the chatbot gives irrelevant or redundant answers, or when hardware notification is too slow, user engagement and trust can dip over time.

- **Notification Accuracy:** Making sure real-time sensor alerts are accurate and timely is vital for user safety. False alarms or failure to send critical notifications can undermine user confidence in the app's monitoring abilities.

2.4.3 Social and Cultural Barriers

- **Resistance to Smart Systems:** Certain users might opt for conventional meal planning practices or feel reluctant to rely on AI-based systems and IoT devices for safety monitoring. Promoting adoption demands showcasing the value, reliability, and simplicity of the system through education and positive usage experiences.

- **Diverse Expectations:** The app has to adapt to a broad range of users who have varying food options, safety issues, technological familiarity levels. Designing flexible, culturally sensitive, and easy-to-use features is necessary to make the app accessible and appealing to a global user base.

2.5 Real-Time Monitoring and Feedback System

Monitoring and feedback are essential parts of intelligent applications, providing insights into user behavior, system effectiveness, and safety measures. In the Smart Meals App, real-time feedback mechanisms provide timely meal suggestions and environmental alarms to the users, enhancing overall user experience and confidence in the system.

2.5.1 Types of Monitoring and Feedback

- **Real-Time Sensor Monitoring:** Real-Time Sensor Monitoring: Hardware integration within the app constantly monitors environmental factors like temperature, gas levels, and sound detection. When any unusual readings are found, the system automatically sends notifications to users, enabling them to react in a timely manner to avert accidents.

- **Feedback through Chatbot:** The chatbot integrated with AI delivers real-time responses to user inquiries regarding meals, ingredients, or health information. This real-time feedback enables users to make informed choices at a fast pace and leads to regular interactions with the app.

2.5.2 Technological Innovations in Assessment

- **Alert systems through Automated Systems:** The use of Firebase and ESP32 microcontrollers facilitates real-time, automatic notification of the user's mobile device. When a sensor senses threshold violation (e.g., excessive heat, gas leak), the system automatically initiates alerts and even executes automatic actions such as switching on an exhaust fan.

- **Intelligent Response Optimization:** Leverage Retrieval-Augmented Generation (RAG) methods to make responses more personalized based on user interaction history, enhancing relevance and quality of response over time. This customized response creation improves user satisfaction and ensures the feedback is accurate and useful.

2.6 Smart Kitchen Literature Survey and Limitations

A number of literature reviews on smart kitchen systems, AI-powered meal aids, and IoT home monitoring demonstrate a quickly developing technological environment. Studies show that AI-powered chatbots significantly enhance user interaction by providing personalized and dynamic support, while IoT devices contribute to real-time environmental monitoring, enhancing household safety. However, several limitations persist, including challenges in ensuring accurate contextual understanding by AI models, reliability concerns regarding real-time sensor communication and the need for continuous adaptation to emerging technological trends. Furthermore, difficulties in seamless integration between machine learning models

and IoT systems, network dependency, and the absence of standardized frameworks for evaluating the effectiveness of AI-IoT combined platforms highlight the complexity involved in successfully developing and deploying such smart systems.

2.6.1 Literature Survey

2.6.1.1 Retrieval-Augmented Generation (RAG) in NLP Tasks

This article [1] discusses an approach that merges pre-trained models and external non-parametric memory (like a dense vector index of Wikipedia) to facilitate improved knowledge access in language generation. Authors compare two distinct methods of retrieval-augmented generation (RAG) and compare their performance on a number of NLP tasks. The findings prove that RAG models perform better than common language models in activities such as open-domain question answering and language generation, generating more precise and varied results.

2.6.1.2 Kitchen Automation for Safety and Control (2022–Present)

A smart safety kitchen system automates kitchen operations like gas leak detection, fire suppression, and temperature control. Utilizing sensors to monitor the kitchen space, the system triggers alarms and safety mechanisms such as gas cutoffs and water sprays when necessary. It can be accessed using a smartphone application, promoting enhanced home security and kitchen automation [2].

2.6.1.3 RAGAS Framework for Personalized Education (2024–Present)

The paper [3] introduces a framework named RAGAS that applies Retrieval-Augmented Generation (RAG) in conjunction with web scraping and language models (such as Llama 3) for effective data retrieval and summarization. The model is very effective in personalized education, where it produces customized learning materials for learners. The results indicate that Llama 3 is most accurate in summarizing data, thus highly ideal for personalized tutoring sessions.

2.6.1.4 RAG-Based IT Support Chatbots (2024–Present)

This article [4] discusses a RAG-based chatbot intended for technical support in an IT consulting company in Canada. By including domain-specific data during retrieval, the chatbot provides more precise and effective replies than general LLM models. The system was tested in a real-world production setting and displayed remarkable improvements in customer satisfaction according to ROUGE scores and user comments.

Table 2.1: Overview of Smart Kitchen Techniques and Limitations

Technique	**Description**	**Example**	**Limitations**
AI Chatbots (2010s–Present)	AI-powered conversational agents providing personalized responses using techniques like Retrieval-Augmented Generation (RAG).	Meal recommendation bots using LangChain and FAISS-based retrieval.	May struggle with highly context-specific or ambiguous queries; risk of outdated knowledge if not updated.
IoT-based Kitchen Monitoring (2010s–Present)	Use of smart sensors (temperature, gas, sound) to monitor the kitchen and living spaces in real time.	Smart exhaust fan control systems triggered by temperature sensors.	Requires stable internet connection; sensor calibration needed for accuracy.
Firebase Realtime Database Integration (Present)	Cloud-based backend enabling real-time data synchronization between IoT hardware and mobile applications.	Firebase alerts triggered by sensor activity integrated with mobile apps.	Dependency on cloud services; potential latency issues under heavy loads.
Retrieval-Augmented Generation (RAG) for Chatbots (Present)	Combines information retrieval from a local knowledge base with natural language generation for intelligent, up-to-date responses.	LangChain-based meal assistance using FAISS vector search.	Requires careful tuning of retrieval parameters; fallback mechanisms needed when local knowledge fails.
Hybrid AI-IoT Systems (Present)	Systems combining AI decision-making with IoT-based physical monitoring for smart environment management.	Apps combining meal assistance with kitchen safety monitoring.	Complex to design and maintain; requires multidisciplinary expertise (AI + hardware).
Descriptive Studies on Smart Home Adoption	Research conducted to understand user acceptance and challenges of AI and IoT technologies at home.	Surveys analyzing smart kitchen system usage during recent years.	Cannot directly infer causality; influenced by individual user biases and experiences.

2.6.1.5 PDF-to-AI Chatbot for Document Interaction (2024–Present)

The PDF-to-AI Chatbot technology translates static PDF documents into dynamic, interactive chatbot conversations. The system enables users to engage with documents in a more interactive manner, retrieving key information and getting customized suggestions. It is accessible on different digital platforms and is meant to make reading documents more affordable and conversational [5].

2.6.1.6 BIM Knowledge Enhancement with RAG for ChatGPT-4 (2024–Present)

This research [6] aimed at improving the performance of ChatGPT-4 on the Korea BIM Expertise Test using Retrieval-Augmented Generation. The results indicated that incorporating RAG considerably increased ChatGPT-4's performance by giving localized and domain-specific documents, proving RAG's capability to enhance specialized knowledge.

2.6.1.7 Automatic Exhaust Fan Systems in Kitchens (2024–Present)

An automatic exhaust fan system with temperature sensors to control kitchen ventilation was established. The system maintains appropriate air quality and cooking temperature control, offering an easy process for ensuring comfort and safety in the kitchen setting [7].

2.6.1.8 AI and ML in Sustainable Manufacturing (2000–Present)

The coming together of AI and ML in manufacturing has been a big step toward making the process sustainable. Both technologies allow for more intelligent process systems that maximize the use of resources and reduce environmental footprint. Literature [8] has demonstrated how the use of AI and ML in manufacturing aligns closely with Industry 4.0, where data-driven decision-making and automation play central roles.

2.6.2 Integration of Smart Kitchen with Industry 4.0

Mobile application integration for Industry 4.0 technology adds personalization, automation, real-time communication, and system intelligence. In the Meals App, technologies like the Internet of Things (IoT), Artificial Intelligence (AI), Big Data Analytics, and Cloud Computing have crucial roles to play in reshaping the user experience and system functionality.

2.6.2.1 Internet of Things (IoT)

- **Smart Monitoring:** Smart Monitoring: Hardware sensor integration like temperature sensors, gas sensors, and sound sensors allows for real-time monitoring of the kitchen space. Information gathered from these IoT devices is

continuously transmitted to the cloud, allowing users to receive immediate alerts and take necessary actions to ensure safety.

2.6.2.2 Artificial Intelligence (AI) and Machine Learning (ML)

- **Personalized Assistance:** Personalized Assistance: The AI-powered chatbot uses Retrieval-Augmented Generation (RAG) techniques to provide personalized meal recommendations based on user queries. Machine learning allows the chatbot to improve its responses over time by learning from previous user interactions, making the experience more adaptive and relevant.

2.6.2.3 Big Data Analytics:

- **Smart Insights:** The Meals App gathers useful user interaction data, sensor event data, and system usage patterns. Analyzing this big data allows the system to optimize future meal suggestions, improve notification accuracy, and enhance the overall system responsiveness, making it more predictive and intelligent.

2.6.2.4 Cloud Computing:

- **Real-Time Communication:** •The app uses Firebase Realtime Database to maintain continuous communication between sensors and the mobile application. Cloud computing ensures scalability, secure data storage, and real-time alert delivery, supporting a seamless user experience regardless of geographic location.

2.6.3 Challenges in Implementing Industry 4.0 in Smart Kitchen

While integrating Industry 4.0 technologies into smart mobile applications brings many advantages, several challenges must be addressed to ensure successful implementation.

- **Infrastructure and Cost:**Developing and maintaining a system that integrates AI, IoT devices, real-time cloud communication, and data analytics requires significant investment in hardware, software development, and infrastructure setup, which may limit scalability for small user bases initially.

- **Data Privacy and Security:**Ensure the security and privacy of sensitive information..
- **Digital Divide**Some users, particularly in underserved areas, may face challenges accessing the full capabilities of the application due to limitations in internet connectivity, availability of compatible smartphones, or familiarity with smart technologies.
- **Technological Complexity:**Complexity in integrating and managing advanced technologies.
- **Resistance to Change:**Users who are not familiar with AI-based systems or IoT technologies may hesitate to adopt new features. Building intuitive, simple user interfaces and providing user education can help overcome this resistance.
- **Interoperability Issues::**Ensuring different systems and technologies work seam- lessly together.
- **Maintenance and Updates:**•Ongoing maintenance and regular updates of technology systems.

2.7 Impact of AI-Powered Smart Applications on User Experience

A number of studies have established that mobile apps powered by AI supplemented with real-time monitoring technologies significantly improve user experience, satisfaction, and engagement. Smart applications like the Smart Meals App illustrate the capability to customize user interaction, enhance security, and provide real-time response, revolutionizing how users handle their daily lives.

2.7.1 Key Findings

- **Enhanced Accessibility:** The Meals App facilitates customized meal planning and kitchen monitoring that is available to users independent of their geographic location. The utilization of mobile platforms and cloud-based technologies such as Firebase means that users across varying backgrounds can have access to real-time safety notices and smart meal recommendations, enabling larger access to smart living solutions.
- **More Powerful User Experience:** The inclusion of an AI-based chatbot utilizing Retrieval-Augmented Generation (RAG) methods allows for

context-aware and dynamic interactions. This tailored support, coupled with real-time kitchen safety monitoring, produces a rich, interactive, and highly responsive experience for users, more efficiently and securely completing daily tasks.

- **Positive User Feedback:** Initial user feedback from users of AI-based applications shows high levels of satisfaction with features such as flexibility, real-time responsiveness, and smart suggestions. Users welcome the option of receiving immediate notifications, conversing with an intelligent AI assistant, and tailoring the app's behavior based to their own needs, resulting in greater long-term participation and trust.

2.8 Case Studies and Best Practices

A number of case studies of AI-driven mobile app development and intelligent IoT systems illustrate effective implementation of tailored support, real-time tracking, and cloud connectivity. These examples provide useful lessons on best practices and strategies for achieving typical technical and user engagement obstacles.

2.8.1 Case Study Examples

- **Google Assistant and Smart Home Integration:** Google Assistant has been able to integrate AI-conversational systems with smart home appliance control so that users can control lights, security systems, and appliances with ease using simple voice commands. Its success lies in seamless integration with IoT devices, real-time responsiveness, and delivering personalized user experiences based on individual preferences.

- **Samsung Smart Things:** Samsung's SmartThings platform connects IoT devices such as sensors, alarms, and smart appliances into a centralized mobile app. The platform demonstrates the power of real-time monitoring, automatic alerts, and user-configurable automation routines, serving as a strong model for integrating hardware control with cloud-based mobile applications.

- **MyFitnessPal App:** MyFitnessPal uses AI to provide personalized diet tracking to users. It highlights the effectiveness of AI in delivering adaptive suggestions based on user input, improving engagement and promoting healthier living habits through continuous feedback and personalized content.

2.9 Gaps in Existing Literature

- **Longitudinal Studies:** There is a need for long-term studies on the impact of AI powered app and real-time monitoring systems on user behavior and safety.
- **Holistic Approaches:** Few solutions fully integrate AI chatbots and IoT monitoring into a single seamless mobile platform. More comprehensive approaches are required.
- **Cultural Sensitivity:**Current applications often lack adaptability to different cultural food habits and languages. More culturally aware designs are needed..
- **System Reliability:**Research on validating the reliability of real-time alerts from sensors in home environments is limited. Studies on trust and false alarms are necessary.
- **Inclusive Design**Many smart apps do not fully address the needs of users with disabilities or limited technical skills. More accessible designs are required.

2.10 Conclusion

This chapter reviewed the key technologies, trends, and research developments related to AI-powered mobile applications and real-time smart monitoring systems. It highlighted how advancements in AI chatbots, IoT integration, cloud communication, and personalized user experiences have shaped the evolution of intelligent mobile solutions. Several successful case studies were discussed, offering insights into best practices and strategies for combining real-time hardware monitoring with AI-driven user assistance. At the same time, various gaps in the existing literature were identified, including the need for long-term impact studies, better cultural sensitivity, system validation, inclusive designs, and cost-effectiveness analyses. The review emphasizes the growing importance of unified, intelligent mobile systems that can personalize assistance while ensuring safety through real-time monitoring. This understanding forms the foundation for the design, development, and innovation introduced in the Smart Meals App.

Chapter 3

Tools and Libraries

In this chapter, we introduce the tools, libraries, and frameworks that constitute the backbone of the Smart Kitchen. These resources were chosen with care to isfy the technical requirements of the project, such as mobile app development, real-time cloud communication, integration of machine learning, and hardware connectivity. Every tool is crucial in the provision of the application's performance, scalability, and user experience.

3.1 Framework

3.1.1 Flutter

- Flutter is an open source UI framework for building natively compiled apps for , web, and desktop from a single codebase.
- It ships with a rich set of customizable widgets, enabling fast and efficient UI with smooth animations and high performance.
- One of the most prominent features of Flutter is its "hot reload" feature, which allows developers to view changes in the code immediately displayed in the application running without rebooting.
- This greatly accelerates the development cycle and improves productivity.
- Flutter employs the use of Dart programming language, where strong typing is coupled with a compact syntax, making it well positioned for both frontend and backend development tasks within the app.
- In Meal App, Flutter is employed to create the user interface, handle navigation between screens, manage real-time Firebase notifications, and engage with the AI-driven chatbot.

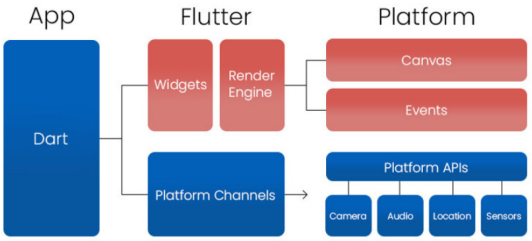

FIGURE 3.1: Flutter Framework

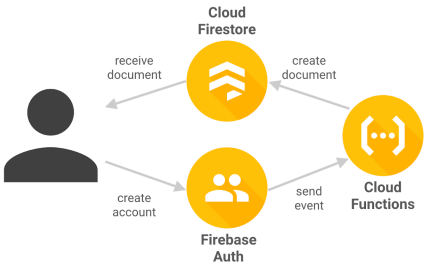

FIGURE 3.2: Firebase Working

Its efficiency and flexibility make it a perfect fit to develop an advanced, responsive mobile app that performs well on both Android and iOS platforms (Figure 3.1).

3.2 Backend Services and Libraries

3.2.1 Firebase

Firebase is a cloud-hosted Backend-as-a-Service (BaaS) platform created by Google, which is very popular for creating real-time and scalable mobile apps. Firebase is a key component in the Smart Kitchen Assistant project with its provision of important backend functionalities like real-time data synchronization, user authentication, and cloud-based storage of data. Its real-time features facilitate real-time syncing between the ESP32-based embedded system and the Flutter-based mobile app, which is essential for real-time kitchen safety notices and repeated user experience (Figure 3.2).

3.2.1.1 Realtime Database

Firebase Realtime Database stores and synchronizes the sensor data of the ESP32 microcontroller in real time. This comprises key kitchen metrics like gas levels, temperature, and sound detection. Sensor value changes are immediately propagated to the app UI, allowing for timely user alerts and automatic responses (such as fan operation on gas leaks).

3.2.1.2 Data Storage and Sync

Firebase acts as a centralized sync layer to store user preferences such as meal favorites, chatbot interaction history, and fan control commands. Writing data to Firebase from either the ESP32 hardware or mobile app is mirrored on all devices in real time. Bi-directional syncing is crucial to make the smart kitchen experience interactive and seamless.

3.2.1.3 Real-time Responsiveness

Firebase's real-time capabilities make real-time notification and UI updates possible within the app. As an example, if gas concentration is beyond a certain level, a real-time alert is sent to the user and the exhaust fan is automatically triggered. This automatic response is central to the safety-first strategy of the Smart Kitchen Assistant.

3.3 AI Chatbot Module

3.3.1 LangChain and RAG Implementation

LangChain is a robust open-source Python library that makes application development with large language models (LLMs) accessible by offering tools for handling inputs, outputs, memory, chains, and interaction with external resources. In the Smart Kitchen Assistant project, LangChain is central role in the deployment of the Retrieval-Augmented Generation (RAG) pipeline, which drives the AI chatbot component. The central notion behind the RAG architecture is to fortify the abilities of the language model by anchoring it with pertinent external information—specifically from domain-related documents. For this project, the backend system is pre-loaded with PDF documents containing recipes, cooking methods, food safety guidelines, and health tips. These are initially processed with the help of the PyPDFLoader utility of LangChain to extract the textual content from each of the PDF files. After receiving the textual content, it goes through the RecursiveCharacterTextSplitter, another LangChain utility, which splits the documents intelligently into smaller chunks, which are semantically significant. This makes sure that each chunk is contextually meaningful as well as of an optimal length for embedding. The processed chunks are then transformed into high-dimensional vector representations using OpenAI's embedding model (text-embedding-ada-002). These vector embeddings are stored in a FAISS (Facebook AI Similarity Search) vector database, which allows for fast and efficient similarity-based document retrieval. FAISS enables the system to match

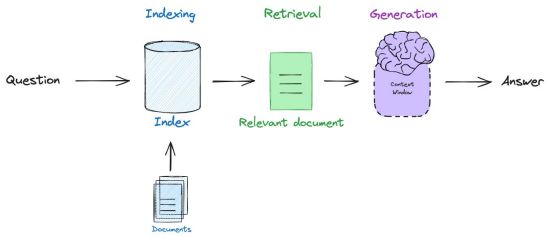

FIGURE 3.3: Basic RAG

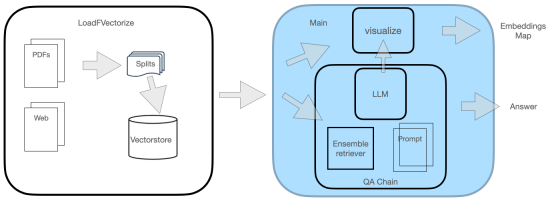

FIGURE 3.4: Working with FAISS

a user's query to the most relevant chunks of text by comparing the cosine similarity of the query vector with those that are in the database. At runtime, when a user poses a question through the mobile application (e.g., "What's a good high-protein breakfast?"), LangChain's ConversationalRetrievalChain acts. The query is embedded first and matched against the stored vectors in FAISS to retrieve the most contextually relevant information. This retrieved information is then appended to the original query and passed to OpenAI's GPT-3.5-Turbo model to generate a coherent, informative, and conversational response. This dynamic combination of document retrieval and language model inference ensures that the chatbot can provide accurate, context-aware answers grounded in verified documents, rather than relying solely on the language model's internal training. In case no appropriate documents are located or the content obtained lacks adequate confidence, the system resorts to making use of the DuckDuckGo Search API to retrieve current information from the web. LangChain overall allows the chatbot of the Smart Kitchen Assistant to function as a trustworthy and smart virtual cooking consultant by smoothly combining document-based retrieval and current language generation inside a modular and scalable infrastructure (Figure 3.3).

3.3.2 FAISS Vector Store

FAISS, created by Facebook AI Research, indexes and retrieves efficiently similar document embeddings. The chatbot sends FAISS user inputs to retrieve the most appropriate document chunks, which serve as context for GPT-3.5 Turbo to produce correct and coherent answers (Figure 3.4).

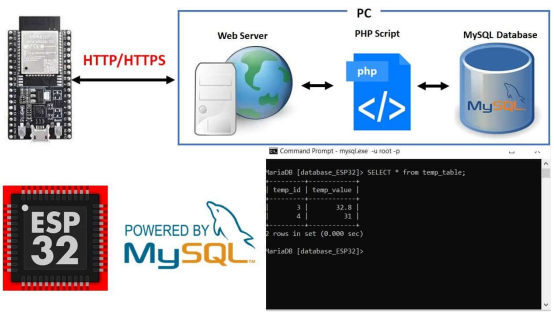

FIGURE 3.5: Working with ESP32 Microcontroller

3.3.3 OpenAI API and DuckDuckGo Search

OpenAI's GPT-3.5 Turbo model generates conversational answers based on retrieved context. When the document context is insufficient, the chatbot falls back to DuckDuckGo's web search API for live internet results, ensuring comprehensive and up-to-date assistance.

3.4 Embedded Hardware and Sensor Modules

3.4.1 ESP32 Microcontroller

The ESP32 microcontroller provides Wi-Fi connectivity and sensor interfacing. It collects environmental data and uploads it to Firebase, enabling real-time monitoring (Figure 3.5).

3.4.2 Sensors

- **MQ2 Gas Sensor**: : It detects flammable gases and smoke to detect any potential danger.
- **DHT11/DHT22 Temperature Sensor**: It senses ambient temperature for safety and comfort tracking. Microphone Sound Sensor: It detects any loud cry by the baby or abnormal sound to notify users
- **Microphone Sound Sensor**: Regulates kitchen appliances like exhaust fans as per sensor input or user inputs.
- **Relay Module**: Controls kitchen appliances such as exhaust fans based on sensor input or user commands.

3.5 Supporting Libraries and Packages

This section points out the different libraries and packages that assist in the development and operation of the Smart Kitchen Assistant app. These packages are grouped according to the platform on which they are used—Flutter for frontend mobile development, ESP32 for embedded firmware, and Python for backend AI chatbot processing.

3.5.1 Flutter Packages

3.5.1.1 google_fonts

This package is utilized to include Google Fonts in the Flutter app, supporting custom typography and improving the aesthetic value of the user interface.

3.5.1.2 provider

The provider package is utilized for state management of the app. It makes data flow and UI updating easier, supporting efficient interaction between screens and components.

3.5.1.3 transparent_image

This package smoothenes image loading by utilizing transparent placeholders during real images are being retrieved and displayed, enhancing user experience during navigation.

3.5.2 ESP32 Firmware Libraries

This section describes the major firmware libraries applied in programming the ESP32 microcontroller, which manages real-time sensor data collection and interaction with the Firebase Realtime Database. These libraries make the microcontroller function smoothingly within a networked system, promoting smart kitchen automation and monitoring.

3.5.2.1 WiFi.h

The WiFi.h library is a crucial library to use when setting up and controlling Wi-Fi connections on the ESP32 microcontroller. It can be used to connect the device to a given Wi-Fi network, thus making the internet accessible. The connection is important for sending sensor readings—gas levels, temperature, and sound—back from the ESP32 to the Firebase Realtime Database. Moreover, it provides the microcontroller to accept control commands from the mobile application, i.e., turning the kitchen exhaust fan on and off. Without this library, the ESP32 would not be able to be part of real-time communication in the smart

kitchen environment.

3.5.2.2 FirebaseESP32.h

The FirebaseESP32.h library is the interface between the ESP32 microcontroller and Google's Firebase. It makes reading from and writing to these sources easier to the Firebase Realtime Database. In the Smart Kitchen Assistant system, this library is employed to upload real-time sensor data (e.g., gas concentration or temperature values) to the cloud. It also makes it possible for the ESP32 to respond to state changes initiated from the mobile app user interface—like receiving the instruction to turn on the exhaust fan. The use of this library provides bi-directional communication between the hardware layer and the cloud, allowing for a timely and interactive system.

3.5.2.3 DHT.h

The DHT.h library is actually intended to be used in interfacing with DHT11 or DHT22 sensors, used to sense temperature and humidity. In the application of this project, the temperature readings made with this library are important in tracking kitchen conditions. Through constant monitoring of the temperature, the system is able to detect overheating situations that can necessitate instant ventilation or operator intervention. The library has easy-to-use APIs for initializing the sensor, reading, and converting raw data to consumable temperature values in either Celsius or Fahrenheit

3.5.2.4 ArduinoJson

ArduinoJson is a widely used C++ library applied in embedded systems for parsing and creating JSON-formatted data. Here, it is used to format the sensor data gathered by the ESP32 before it is sent to Firebase. Given that Firebase functions based on JSON, this library guarantees the data format is compatible and easily comprehensible to both the mobile application and the database. Use of ArduinoJson simplifies communication, minimizes the occurrence of data errors, and improves the scalability of the system by standardizing data exchange protocols

3.5.3 Python Backend Libraries

This section addresses the Python libraries employed in the backend module of the Smart Kitchen Assistant, which drives the Retrieval-Augmented Generation (RAG)-based AI chatbot. These tools collaborate to extract, process, and manage knowledge from cooking-related PDF documents, making it possible for the AI assistant to produce intelligent and context-aware responses to user queries.

3.5.3.1 PyPDFLoader

PyPDFLoader is a LangChain library utility used to load and extract text content from PDFs. Within this project, PDF files with recipes, culinary methods, health tips, and nutrition guides are read into the system with the help of PyPDFLoader. The extracted text is used as a knowledge base by the AI chatbot. Utilizing this tool, the backend system can process complicated, multi-page documents and extract them into machine-readable formats for subsequent processing.

3.5.3.2 RecursiveCharacterTextSplitter

Another LangChain tool that splits big chunks of text into manageable pieces or chunks is the RecursiveCharacterTextSplitter. This is crucial for efficient embedding and retrieval processes. Once the PDFs are loaded using PyPDFLoader, the text is split by this tool so that each segment remains semantically coherent while staying within the token limits required by embedding models like OpenAI's text-embedding-ada-002. These chunks are then indexed in a vector database (FAISS) to support fast and relevant document retrieval during chatbot queries.

3.5.3.3 dotenv

The dotenv library loads environment variables from a .env file into the backend runtime environment. It securely holds sensitive credentials like API keys for the OpenAI model and Firebase settings. This approach to secrets reduces the overall security and maintainability risks of the codebase. Instead of hardcoding credentials, which is risky and error-inducing, using dotenv means private data will be kept isolated from the core application logic, rendering the backend safer as well as more professional in nature.

Chapter 4

Sustainable Development Goals (SDG's)

4.1 Introduction

The United Nations' Sustainable Development Goals (SDGs) are a worldwide call to action to end poverty, protect the environment, and ensure peace and prosperity for all. The 17 goals are a basis for solving today's most serious global issues. This chapter examines how our online application aligns with and contributes to some of the key SDGs, demonstrating our commitment to driving sustainable development through technological innovation.

4.2 Good Health and Well-being (SDG-3)

Objective: To ensure healthy lives and well-being for all at all ages. Our Smart Kitchen Assistant project improves health and well-being by enhancing household safety and care through smart technologies. Our main contributions are:

Our Smart Kitchen Assistant project contributes to better health and well-being by improving household safety and care using smart technologies. Key contributions include:

- **Gas Leak Detection:** Gas sensors in the system detect potentially hazardous leaks and immediately alert users, avoiding dangerous exposure.
- **Temperature Monitoring:** Real-time temperature readings avoid overheating in the kitchen, lowering fire or burn risks.

- **Baby Noise Detection:** Recognizes unusual baby sounds while cooking, improving child care and maintaining peace of mind for caregivers.

4.3 Responsible Consumption and Production (SDG-12)

Objective: Ensure sustainable consumption and production patterns. This project encourages intelligent and effective cooking habits and prevents food and energy loss. It makes the following contributions:

This project promotes smart and efficient cooking practices while minimizing food and energy waste. Contributions include:

- **Intelligent Recipe Assistance:** The AI bot recommends dishes based on accessible ingredients, avoiding food waste.
- **Efficient Energy Use:** Automation of fans and safety triggers regulate electricity consumption during cooking.
- **Guided Cooking:** Clear, interactive instructions for meal preparation enable users to cook effectively and prevent resource overuse.

4.4 Sustainable Cities and Communities (SDG-11)

Objective: Make cities and human settlements inclusive, safe, resilient, and sustainable.
The Smart Kitchen Assistant supports the development of smart homes by increasing household safety and convenience. Major contributions include:

- **Real-time Hazard Alerts:** Notifies users of gas leaks, extreme heat, or loud noises to prevent domestic accidents.
- **User-Friendly Dashboard:** Makes safety monitoring accessible to all users, promoting inclusive smart living.
- **Automation Features:** Automatic fan activation enhances safety and efficiency in managing kitchen environments.

4.5 Industry, Innovation, and Infrastructure (SDG-9)

Objective: Build robust infrastructure, promote inclusive and sustainable industrialization, and encourage innovation.
Our project exemplifies modern innovation by combining AI, IoT, and cloud technologies in a household setting. Key features include:

- **Smart Integration:** Combines ESP32 sensors, Firebase, Flutter, and RAG-based AI chatbot for a seamless user experience.
- **Cutting-edge Technology:** Utilizes retrieval-augmented generation (RAG), enabling smart and dynamic meal suggestions based on user queries and preferences.

4.6 Partnership for Goals (SDG-17)

Objective: Strengthen implementation mechanisms and revive the global collaboration for sustainable development.
Our application reflects collaborative efforts in integrating multiple technologies and working with platforms and tools. Key initiatives include:

- **Technology Integration:** Utilized Flutter, Firebase, LangChain, FAISS, and ESP32 sensors through cross-platform collaboration.
- **Open Knowledge Approach:** The system design and modular architecture promote future enhancements and encourage collaborative research or product extension.
- **Community Feedback:** Regular testing and user feedback ensured that the system remains user-centric and impactful.

4.7 Conclusion

Our Smart Kitchen Assistant adds value to a number of Sustainable Development Goals by applying modern AI and IoT technologies to improve household safety, efficiency, and accessibility. This project illustrates how smart systems can contribute constructively to everyday life and cooperate with international efforts to build a healthier, more sustainable future. With ongoing improvement of the project, we continue to be committed to responsible innovation and inclusive development.

Chapter 5

Methodology

5.1 Introduction

This chapter presents the systematic approach taken to developing the"Smart Kitchen Assistant – A Flutter-Based Application with AI Chatbot and Realy Time Hardware Safety Monitoring." It outlines the planned process from planning and design through implementation, testing, and deployment, with particular focus on integrating a RAG-powered AI chatbot—a novel and core feature of the system. The approach was aimed at developing a dependable, scalable, and intuitive app that boosts kitchen safety and meal support using state-of-the-art AI and IoT technologies.

5.2 Development Process

Development was a pursuit of the Agile model, where incremental progress was possible, independent feedback was afforded at all stages, and flexibility was maintained in accepting change. The goal was effectively achieved through this method in providing incremental enhancements and quick adaptation, especially in sharpening the RAG chatbot and real-time monitoring capabilities.

5.3 Planning

Requirements were collected in the planning phase using stakeholder meetings and informal polls among the target users such as homemakers, cooking fans, and tech-savvy people.

- **Requirement Gathering:** Initial interviews and polls identified the requirement for a smart system providing meal ideas and kitchen safety monitoring. Feedbacks were gathered on desired features such as recipe guidance,

gas/smoke detection, and baby sound sensing.

- **Project Roadmap:** Major development stages and milestones were planned over a specific period of time. Deadlines were established for implementing the vision of integrating the AI chatbot, hardware communication, and user terface improvements.
- **Prioritization:** The key features such as the AI chatbot, hardware monitoring, and Firebase synchronization were given precedence because the impact is immediate on usability and safety.

5.4 Design

The application's interface and architecture were designed using modern UI/UX practices and diagrammatic representations to ensure intuitive navigation and robust system flow.

- **Wireframes and Mock-ups:** Low-fidelity sketches were developed and refined into interactive prototypes using Figma. These helped visualize user journeys and feature placement.
- **Stakeholder Feedback:** Regular review meetings were conducted with faculty and peers to gather insights and suggestions on UI/UX refinements.
- **UI/UX Principles:** Simplicity, responsiveness, and accessibility guided design decisions to create an inclusive and smooth user experience.

5.4.1 Database Design

Firebase Realtime Database was selected for its ease of use with Flutter and real-time syncing capabilities.

- **Structure:** Collections were designed for user profiles, sensor data logs, and chatbot queries.
- **Security Rules:** Firebase's built-in authentication and rules ensured data privacy and secure access control.

5.4.2 API Design

The application uses APIs for integrating AI (via LangChain and DuckDuckGo API) and for sensor data transmission from ESP32 devices.

- **Routing:** Custom endpoints were created to handle:
  - `meals`: Retrieve categorized meal and recipe data.
  - `sensors`: Fetch real-time gas, temperature, and sound data.
  - `chatbot-query`: Process user queries through the AI assistant.
- **Integration Layer:** Middleware in Dart (for mobile side) and Python (for backend chatbot logic) handles input parsing, query generation, and response formatting.
- **RAG Integration:** Middleware processes user inputs and connects them to the LangChain framework for response generation.

5.5 Implementation

The application was implemented using Flutter for cross-platform compatibility, Firebase for backend services, and ESP32 for IoT integration.

- **Backend Development:** Python was used to implement the RAG pipeline using LangChain, FAISS vector database, and the DuckDuckGo API. Middleware was developed to connect the chatbot backend to the mobile interface. Firebase handled user authentication and data storage. Python scripts processed sensor input and triggered alerts.
- **Frontend Development:** Developed in Flutter, the app provides a responsive UI with features for meal browsing, chatbot interaction, and live monitoring of gas, temperature, and sound data from ESP32 sensors.

5.6 Testing

Multiple testing layers were employed to ensure reliability and robustness.

- **Unit Testing:** Each component, including the chatbot response engine and sensor data handlers, was individually tested using mock data.
- **Integration Testing:** Ensured that RAG responses were correctly fetched and returned via the chatbot interface, and that Firebase synchronized hardware data accurately.
- **User Acceptance Testing:** Beta testing was done with home users who provided feedback on the chatbot's accuracy, app usability, and alert responsiveness.

5.7 Deployment

The last step was to transfer the application to a tangible environment common to the general populace.

- **Server Setup:** Firebase Hosting and Functions were used for backend deployment, ensuring scalability and easy access to cloud services.
- **Continuous Integration/Continuous Deployment (CI/CD):** GitHub Actions were integrated to automate build and deployment steps, minimizing manual errors and supporting quick iterations.
- **Monitoring:** Firebase Analytics and logging tools were integrated for monitoring system usage and crash reporting.

5.8 Architecture

The system was designed to ensure modularity, expandability, and ease of maintenance (Figure 5.1).

5.8.1 System Architecture Diagram

The architecture includes the following key components (Figure 5.1):

- **Flutter Frontend:** Handles UI, user inputs, and real-time alerts.
- **Firebase Backend:** Manages authentication, data storage, and real-time synchronization.
- **ESP32 Sensors:** Collect environmental data from the kitchen.
- **Python Backend:** Executes the RAG chatbot logic and integrates with external APIs.

5.9 Data Flow

Understanding the data flow within the application is crucial for ensuring data integrity and efficient processing. The data flow diagram illustrates how data moves through the system, from user input to storage and back to the user.

5.9.1 Data Flow Diagrams

5.9.1.1 DFD Level 0

Shows high-level user interactions — from querying meals to receiving alerts (Figure 5.2).

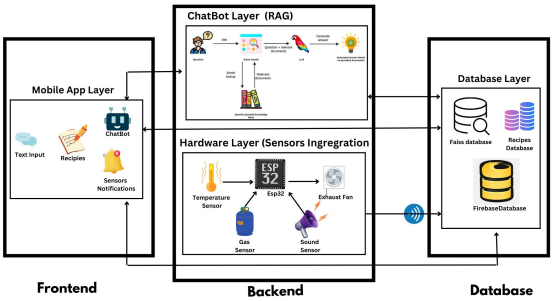

FIGURE 5.1: Architecture Diagram

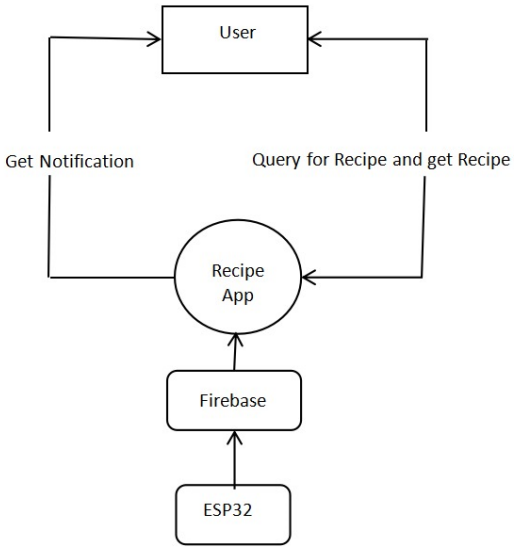

FIGURE 5.2: DFD Level 0

5.9.1.2 DFD Level 1

Breaks down the system into three modules: meal browsing, hardware monitoring, and AI chatbot handling (Figure 5.3).

5.9.1.3 DFD Level 2

(Figure 5.4).

5.10 Use Cases

Use case diagrams help in understanding the interactions between users and the system. They outline the various functionalities the system offers and how different users interact with these functionalities.

5.10.1 Use Case Diagram

The use case diagram represents major interactions like recipe querying, safety alert viewing, and profile management (Figure 5.5).

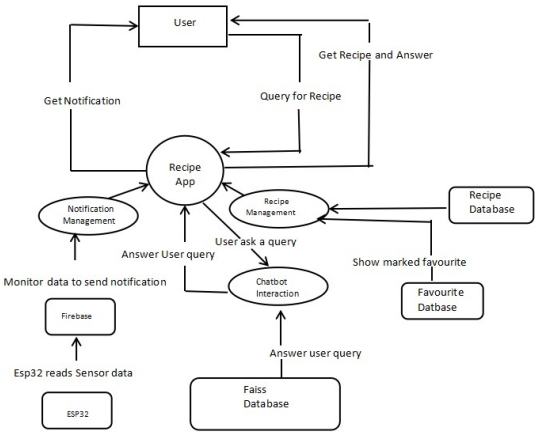

FIGURE 5.3: DFD Level 1

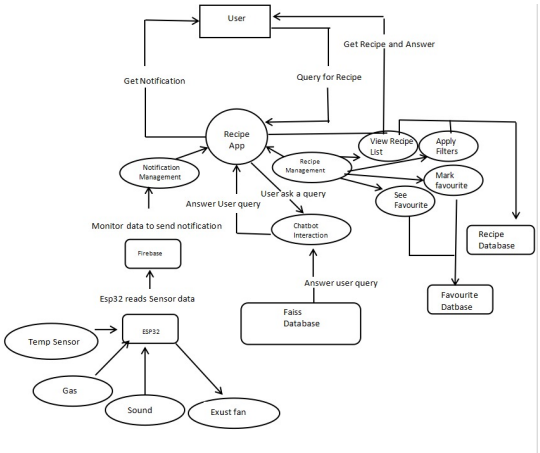

FIGURE 5.4: DFD Level 2

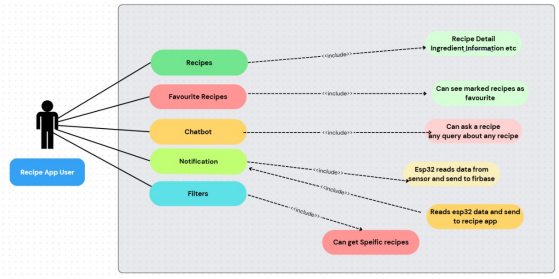

FIGURE 5.5: Use Case Diagram

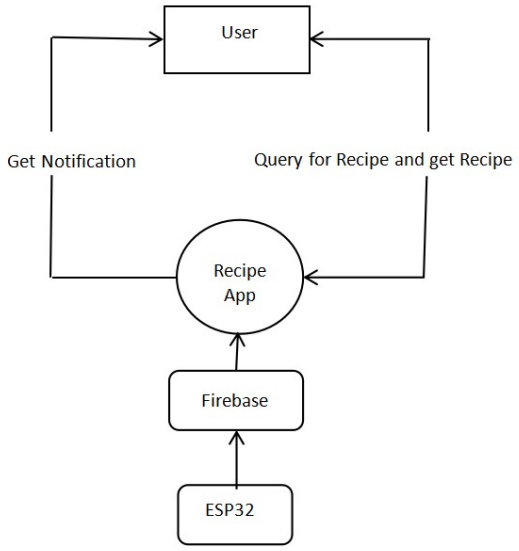

FIGURE 5.6: Flow Diagram (User)

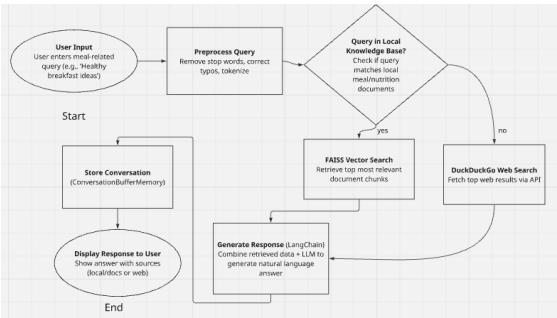

FIGURE 5.7: Flow Diagram (ChatBot)

5.11 User Flows

Understanding the user flows for different types of users is essential for designing an intuitive and effective user experience. The following sections detail the flow for students and instructors.

5.11.1 General User Flow Diagram

Demonstrates a typical journey from login, accessing recipes, viewing sensor alerts, and interacting with the AI chatbot (Figure 5.6).

5.11.2 Chatbot Flow Diagram

The chatbot works in the way shown in (Figure 5.7):

5.11.3 Sensor Flow Diagram

(Figure 5.8).

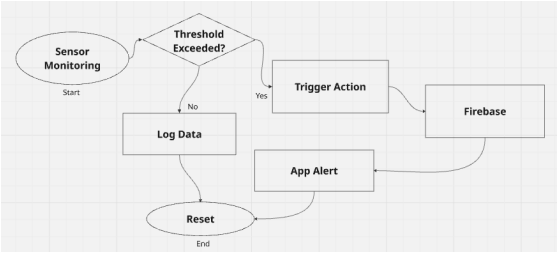

FIGURE 5.8: Flow Diagram (Sensors)

5.12 Challenges and Solutions

It is also essential to know that participants experienced several difficulties when implementing different stages of the development process: technical contradictions, integration issues, and performance problems. This section elaborates on these challenges and the measures undertaken to alleviate them.

5.12.1 Technical Challenges

- **Issue:** One of the primary technical challenges was ensuring smooth handling of complex data relationships between meals, users, and sensor logs in a NoSQL environment like Firebase. Unlike traditional relational databases, Firebase lacks inherent support for relational data modeling, making it difficult to manage associations cleanly.

- **Solution:** To manage this complexity, a structured collection-subcollection schema was used, alongside consistent document referencing. For instance, every user document has pointers to their stored meals or notifications. Attentive normalization, together with Firestore security rules, ensured the data model remained scalable, secure, and efficient.

5.12.2 Integration Challenges

- **Issue:** One of the integration challenges came in synchronizing third-party APIs such as the DuckDuckGo Search API for meal queries, LangChain for RAG-based AI response generation, and real-time sensor feeds from ESP32 devices. Also, making seamless connectivity among these systems without performance loss was paramount.

- **Solution:** A middleware layer was implemented utilizing Dart (on the client side) and Python (on the backend) to coordinate this integration. The chatbot queries were serviced through LangChain pipelines, and the ESP32 data

was passed through HTTP/MQTT to Firebase. Each module was unit-tested and verified to guarantee that integration points were reliable, robust, and asynchronous to avoid blocking operations.

5.12.3 Performance Bottlenecks

- **Issue:**There were performance issues, particularly when dealing with concurrent operations like retrieving meal data, processing chatbot answers, and getting real-time sensor alerts. It sometimes resulted in UI delays or lag.
- **Solution:** : To counteract this, asynchronous programming techniques were implied using Streams and FutureBuilders in Flutter. Data listeners were also optimized to minimize unwanted UI rebuilds. Backend operations were improved by caching frequently used data and through lazy loading mechanisms. The Firebase Realtime Database's ability to selectively observe data changes also minimized unnecessary network requests, enhancing overall responsiveness.

5.13 Conclusion

This chapter introduced a systematic methodology emphasizing the Agile-led development, RAG-augmented AI integration, and real-time IoT monitoring for our "Smart Kitchen Assistant." The project focused not only on innovative AI-enabled meal instruction but also safety through sensor notification. The development process—from planning to deployment—mirrors a high focus on user requirements, smart features, and scalable architecture.

Chapter 6

Implementation

6.1 Development Process

6.1.1 Product Backlog Creation

A product backlog was established to detail the key features of the system, including (Figure 6.1):

- Recipe browsing and categorization
- RAG-powered AI chatbot for recipe guidance
- Real-time IoT-based kitchen safety monitoring (temperature, gas, sound sensors)
- Firebase integration for real-time data synchronization and authentication
- Automatic fan activation and baby noise alert
- Mobile app interface using Flutter

6.1.2 Sprint Planning

Development was divided into sprints. The initial sprint focused on setting up the Flutter frontend and Firebase backend. Subsequent sprints integrated AI chatbot functionality, sensor data handling, and real-time alerts.

6.1.3 Daily Stand-ups

Short daily meetings were conducted to review progress, assign tasks, and resolve technical issues. Collaboration among frontend, backend, and AI integration teams ensured smooth development.

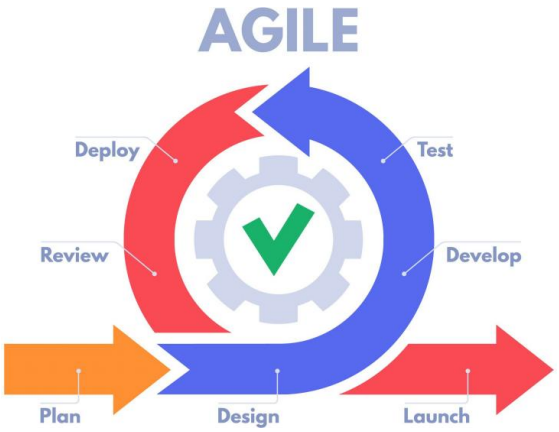

FIGURE 6.1: Agile Development

6.1.4 Sprint Execution

Develop and integrate the selected features, following Agile principles such as incremental delivery and frequent feedback. This may involve coding, designing user interfaces, creating content templates, and setting up hardware integration.

6.1.5 Sprint Review

At the end of each sprint, the current build was demonstrated. Stakeholder feedback, particularly from users in simulated kitchen environments, guided improvements. Gather feedback on the implemented functionalities and adjust priorities or requirements as needed.

6.1.6 Sprint Retrospective

Reflect on the sprint's successes and challenges during the retrospective meeting. Discuss ways to improve processes, collaboration, and product quality for future sprints.

6.1.7 Repeat

Iterate through subsequent sprints, gradually adding new features, improving existing ones, and refining the Smart Kitchen with AI Chatbot and Real-Time Hardware Safety Monitoring based on feedback and evolving requirements.

6.2 Development Technologies

1. **Flutter:** Used to develop a cross-platform mobile application with a clean and responsive UI for recipe browsing, chatbot interaction, and hardware alerts.

2. **RAG (Retrieval-Augmented Generation):** Implemented for the AI

chatbot using LangChain, FAISS, and DuckDuckGo API, enabling dynamic recipe retrieval and conversational guidance.

3. **Firebase:** Enabled real-time data sync, user authentication, and cloud storage for sensor values and user actions.

4. **ESP32 with Sensors:** DHT11 temperature sensor, and a sound sensor (microphone). These sensors collected real-time environmental data from the kitchen, which the ESP32 transmitted continuously to Firebase. This setup enabled live monitoring of gas leaks, temperature changes, and noise levels, enhancing the safety features of the Smart Kitchen Assistant.

5. **LangChain & FAISS:**: To build vector-based document retrieval, enhancing contextual accuracy of chatbot response.

6. **API Integration:** Utilizing DuckDuckGo Search API and external recipe data was to bolster the knowledge base of the chatbot.

7. **Visual Studio Code:** For Flutter and Python coding with extensions for Dart, Firebase, and AI tools.

8. **Git & GitHub:** Source control system utilized for versioning and team collaboration.

6.3 AI Chatbot with RAG-Based Architecture

The chatbot embedded in the Smart Kitchen Assistant employs a RetrievalAugmented Generation (RAG) architecture that provides intelligent, context-specific responses to users. This sophisticated design blends semantic search functionality with high-performance language generation, providing personalized and relevant support, especially for recipe queries.

6.3.1 RAG Architecture

- At the center of the chatbot architecture is the RAG framework that merges vector similarity search with large language model (LLM) generation. FAISS (Facebook AI Similarity Search) is used by the system for efficient semantic search over a curated knowledge base of documents, like recipes and tips on how to cook.

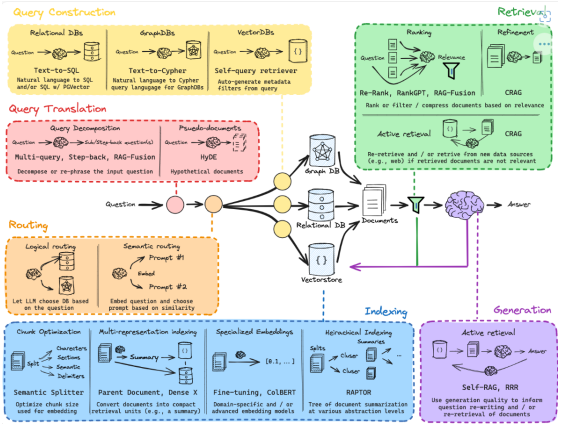

FIGURE 6.2: RAG Architecture

- If the user runs a query, say a query for a recipe based on some ingredients, the system initially converts the query into vector representation. FAISS will then find the top documents from the indexed recipe database based on semantic similarity with the user's query. These documents are used as context and fed as input together with the original query into a large language model through the LangChain framework. coordinates the interaction between the retrieval module and the LLM to make the language model utilize the most applicable external information for providing accurate, coherent, and personalized answers. This strategy greatly improves the chatbot's capability to provide accurate and contextually relevant answers instead of just using pre-trained language models (Figure 6.2).

6.3.2 Recipe Assistance

- One of the major functions of the chatbot is to help users finding and cooking meals. Users can enter questions citing available ingredients or requested types of cuisine, and the chatbot uses the RAG system to retrieve corresponding recipes from the knowledge base (Figure 6.10). retrieving applicable recipe documents, the chatbot recombine the information to present concise, step-by-step cooking instructions and accompanying ingredient lists. This dynamic generation enables the assistant to adjust recipes based on the user's individual limitations, like food preferences or ingredient inclusion, providing useful and easy-to-follow cooking instructions.

6.3.3 Contextual Awareness

- To ensure natural and unforced interactions, the chatbot is contextually aware throughout a conversation. This implies it can recall past messages, monitor user preferences, and answer follow-up questions without losing continuity.

- For instance, when a user first asks for vegetarian recipes and then narrows down request to include preparation time constraints, the chatbot integrates this context to refine its responses. This multi-turn conversational capability enhances user experience by enabling seamless, dialogue-based assistance rather than disjointed, single-query interactions.

6.4 IoT-Based Kitchen Monitoring System

The IoT-based kitchen monitoring system in the Smart Kitchen Assistant application serves as a critical safety feature, enabling real-time detection of dangerous situations in the kitchen environment. This system utilizes the ESP32 microcontroller coupled with a set of sensors—DHT22 for temperature, MQ-6 for gas leak, and a sound sensor to identify abnormal audio patterns like the sound of a baby crying. All data from the sensors is constantly checked and synchronized with Firebase to enable real-time alerts, cloud storage, and user notifications via the mobile app. the overall hardware system is shown in (Figure 6.3).

6.4.1 Temperature Monitoring

- The temperature within the kitchen is monitored continuously with the help of the DHT22 temperature and humidity sensor. The sensor can sense sudden changes in ambient temperature, and this is important in avoiding fire risk overheating conditions

- Within the app, when the temperature in the kitchen exceeds a pre-defined 50°C, the system sends out a warning through the mobile dashboard interface. The real-time alert system makes sure that users are notified on time regarding abnormal heat levels so that they can take timely action to reduce any risk.

6.4.2 Gas Detection

- The MQ-2 gas sensor is included in the system to sense the presence of combustible gases like LPG, methane, and propane. The sensor measures

gas concentration in the environment at all times and reports the readings to Firebase, where they are tracked in real time

- If the gas levels exceed a critical threshold, the system triggers immediate alerts. These alerts are displayed on the sensor monitoring dashboard (see Figure 6.4), and push notifications are sent to the user's device. This early warning mechanism is designed to prevent dangerous accidents such as gas leaks, which could lead to explosions or poisoning if undetected.

6.4.3 Sound Detection (Baby Crying)

- The sound sensor (microphone) is utilized to detect high-pitched or loud sounds, with a specific focus on recognizing the crying of a baby. This feature is especially helpful in households where children are present, allowing parents to be notified immediately in case the child is crying while they are occupied in another part of the house.

- The sensor continuously monitors audio input, and if the sound level exceeds the configured threshold for an extended period, the system interprets it as a potential baby cry. Upon detection, a push notification alert is sent to the user via the mobile app, ensuring that caregivers can respond quickly.

6.4.4 Automatic Fan Control

order to provide improved safety and mimic actual preventive measures, the system has an automatic exhaust fan control. Though used as a simulation during the prototype stage, the logic itself is functional. case the temperature or gas concentration crosses its respective safety threshold, the ESP32 triggers a notification to turn on the exhaust fan. The action is recorded in Firebase and indicated in the application dashboard. This feature is to illustrate the way the system would automatically reduce hazards by enhancing ventilation in an actual usage, thereby avoiding the accumulation of heat or gas in the kitchen setting.

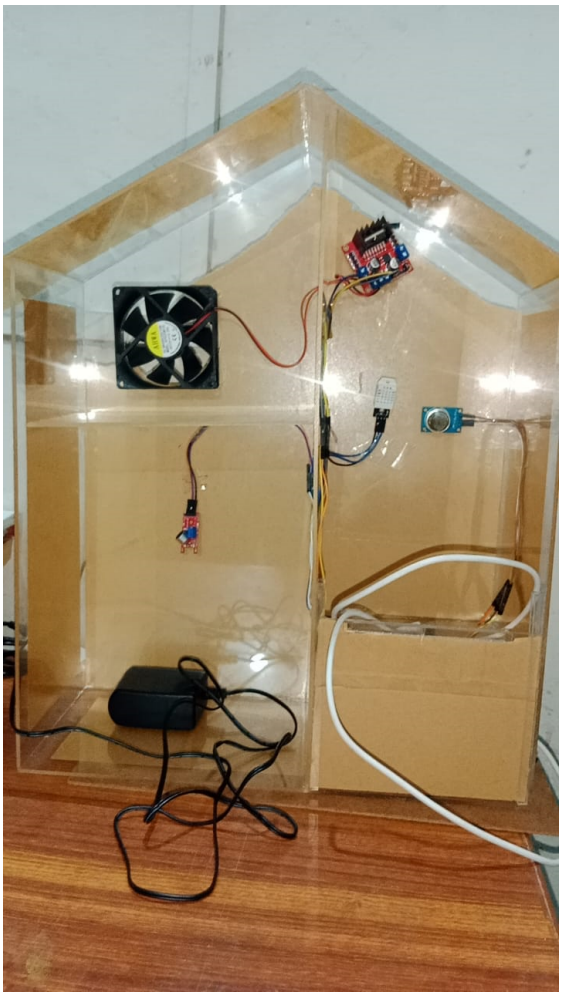

FIGURE 6.3: IOT based Kitchen Monitoring

6.5 User Interface and Experience

The user interface (UI) of the Smart Kitchen Assistant mobile application was created and developed with the use of the Flutter framework. Particular care was taken wards accessibility, seamless interaction sequences, and a streamlined, contemporary look to promote usability by a wide user base. The user interface is responsive, interactive, and facilitates real-time comm- unication with the Firebase-powered backend. Users can find it easy to transition between functionalities like navigation, conversational chatbot interactions,and real-time kitchen safety monitoring.

6.5.1 Recipe Catalog and Navigation

The Recipe Catalog is a central screen within the application, presenting a structured and categorized view of available recipes. The recipes are organized under categories such as (Figure 6.4):

FIGURE 6.4: Categories Screen

- Cuisine Type (e.g., Indian, Italian, Continental)
- Dietary Preferences (e.g., Vegetarian, Vegan, Gluten-Free)
- Cooking Duration (e.g., Under 15 Minutes, Under 30 Minutes)
- Each recipe entry contains a thumbnail image, name, short description, and cooking time (Figure 6.6). Navigating to a recipe takes the user to a view with a list of ingredients and step-by-step cooking instructions (Figure 6.8).

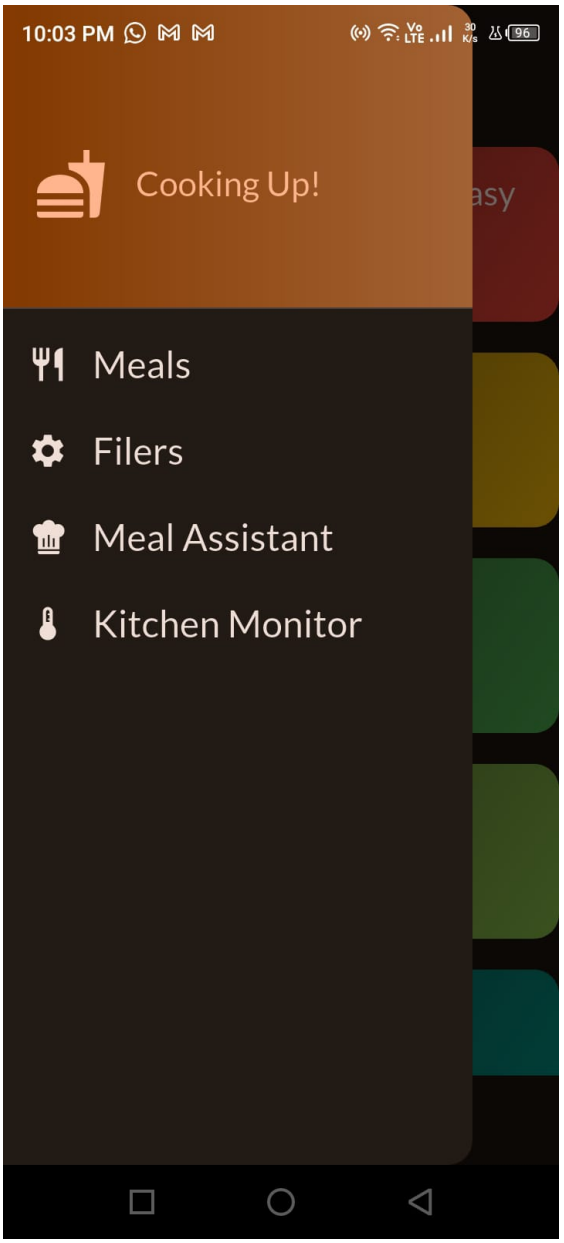

FIGURE 6.5: Navigation Bar

- The in-app search functionality allows users to search by ingredients or names. As users input, results are retrieved dynamically from Firebase without a page reload, promoting a responsive and smooth user experience

6.5.2 Favorites Screen

- Favorites Screen enables customers to mark and quickly retrieve their favorite or most used recipes (Figure 6.7). Every recipe in the portfolio has a "heart" next to it, which may be touched to add/remove it from the list of favorites. This feature increases user convenience as it makes quick recall of favorite recipes possible without repeated searching.

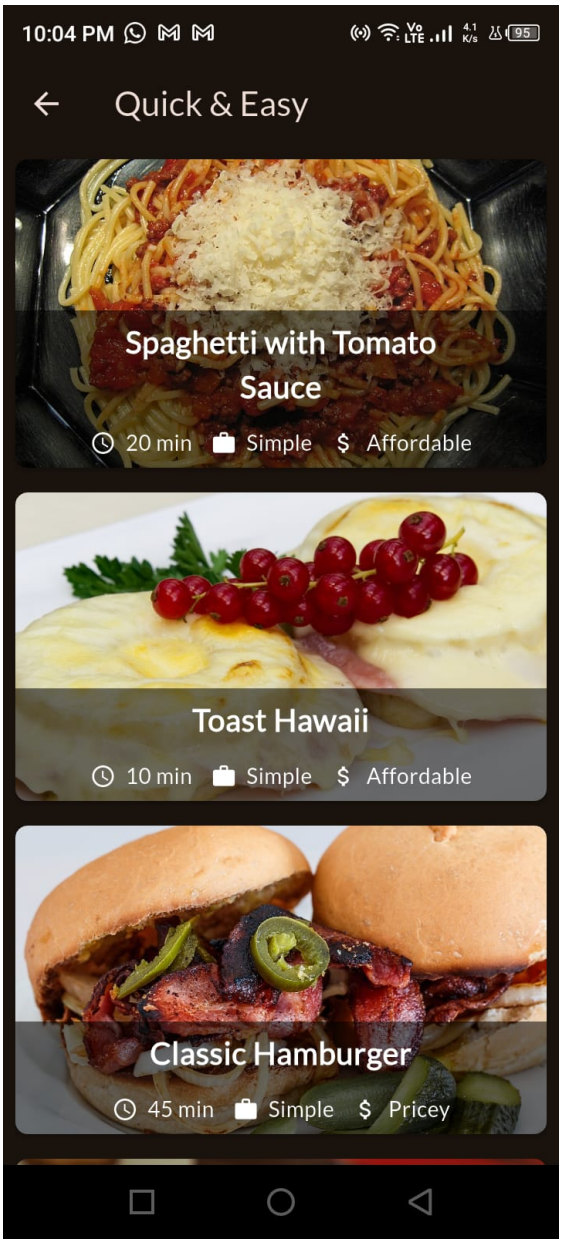

FIGURE 6.6: Available Recipe List

- The screen displays all favorited recipes in a grid or list with the same visual structure applied to the main catalog. Any item can be tapped by the user for ing full recipe information or deleting it from favorites upon a single tap.

6.5.3 Filter Screen

The Filter Screen enables users to refine their recipe searches with advanced filtering options (Figure 6.9). It supports multiple selection criteria, including:

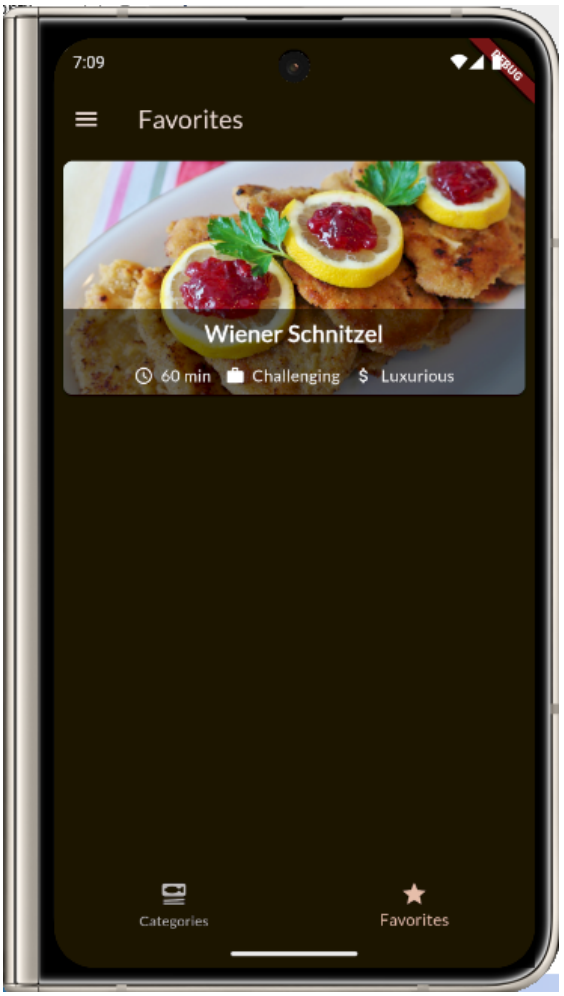

FIGURE 6.7: Favorites Screen

- Dietary Requirements: The Filter Screen allows users to narrow down their recipe searches using advancedfiltering features. It accommodates multiple choice criteria, including:
  - Dietary Requirements: With the consideration of the necessity for health-conscious and lifestyle-orientated eating, the app presents a number of dietary filters. Users can select from:
  - Gluten-Free: Recipes avoiding ingredients that contain gluten.
  - Lactose-Free: : Recipes accommodated for those with lactose intolerance.
  - Vegan: Plant-based recipes without any animal products.
  - Vegetarian: Recipes which do not contain meat but can have dairy or eggs.

FIGURE 6.8: Recipe Details Screen

6.5.4 Chatbot Screen

Chatbot Screen is one of the fundamental features of the Smart Kitchen Assistant app. It offers users an interactive, AI-driven experience to help with recipe discovery, ingredient substitution, cooking instructions, and general kitchen-related questions (Figure 6.10). This screen is home to the RAG-based (Retrieval-Augmented Generation) chatbot, which integrates semantic search with state-of-the-art natural language understanding using a large language model (LLM). The chatbot is intuitive and conversational in nature, resembling the actions of a personal kitchen companion.

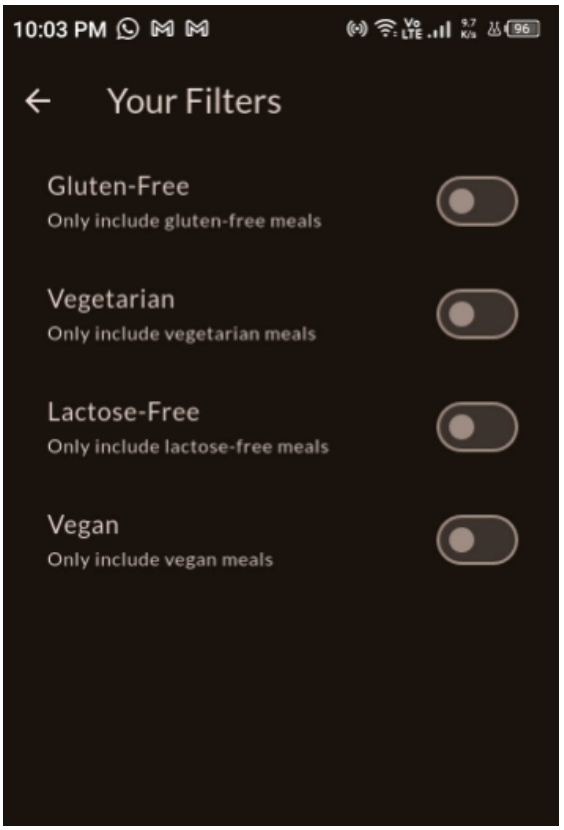

FIGURE 6.9: Filters Screen

Key Features:

- **Text Input and Messaging Interface:** Users can enter natural language questions like "What can I make with potatoes and cheese?" or "How do I make vegan pasta?". The layout of the messages facilitates back-and-forth discussion for smooth interactions.

- **Contextual Awareness:** The chatbot has memory throughout the user turns, enabling follow-up queries such as "Can you reduce the spiciness?" or "Demonstrate a faster version." This increases continuity and user satisfaction

- **RAG Integration:** When a user requests a query, the system fetches the most able recipes or documents from a FAISS-powered vector store. These are then sent to the language model via LangChain to produce a personalized and context-aware response.

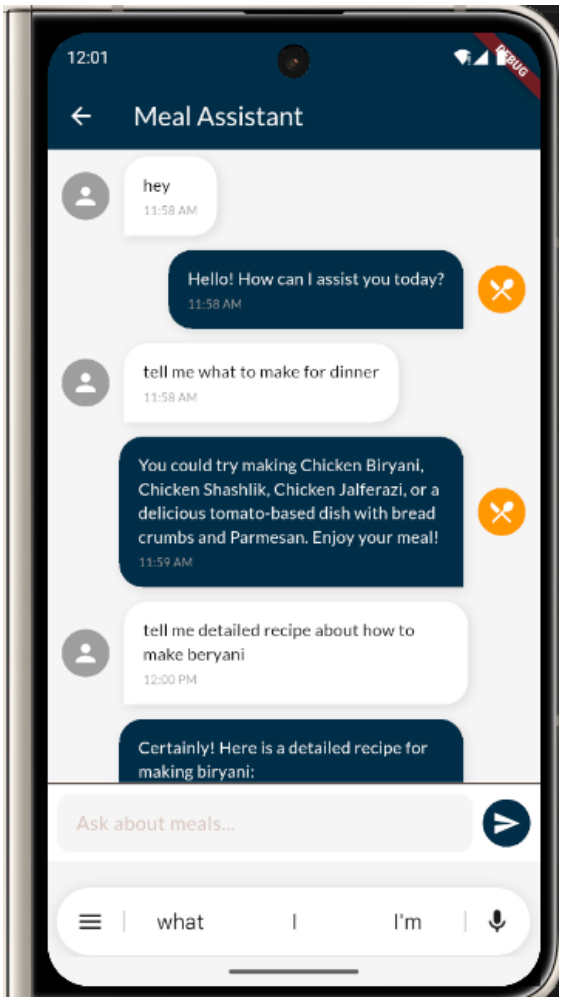

FIGURE 6.10: ChatBot Screen

- **Result Presentation:** Responses contain structured recipe steps, ingredient lists, and optional tips, all presented in a clean, readable format. Hyperlinks or rapid action buttons (e.g., Save to Favorites, View Recipe) are also offered for easy convenience.

The Chatbot Screen improves overall user experience by offering intelligent, human-like kitchen guidance. It serves not just as a recipe recommender but as a wise cooking guide that can cope with multifaceted user needs in real-time.

6.5.5 Real-Time Alert Notifications

Besides the recipe functionalities, the user interface features important alert functionalities associated with the IoT-based kitchen monitoring system. Whenever any abnormal state is sensed—like high gas levels, high temperatures, or loud sounds representing crying of a baby—the app sends instant alerts in plain view of the user (Figure 6.11)(Figure 6.12)(Figure 6.13).

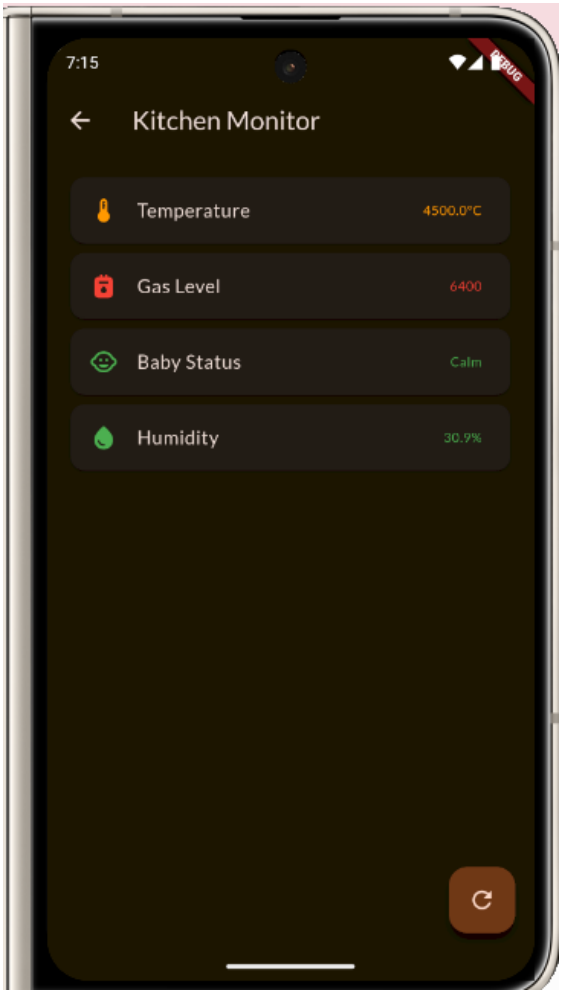

FIGURE 6.11: Sensors Readings

Two types of alerts are sent by these alerts:

- Toast messages: Brief, in-app messages that flash briefly at the bottom of the screen to notify users of existing sensor alarms.
- Push notifications: More intrusive notifications pushed directly to the user's mobile device, guaranteeing alerts even when the application is running in the background or minimized.
- The alert screen offers a real-time display of existing sensor readings and emphasizes any current danger levels. The use of visual indications like color-coded warnings (red for danger, blue for caution) and icons allows users to readily determine the alert type and level of urgency.

FIGURE 6.12: Gas Leakage Notification

6.6 Data Handling and Storage

This chapter describes the different technologies employed in the Smart Kitchen Assistant app for effective, real-time, and smart data processing.

6.6.1 Firebase Firestore

is employed to store structured data like user profiles, recipes, and historical activity logs of sensors. Its real-time indexing and querying are best suited for rapid data retrieval, particularly for recipe browsing and chatbot queries.

6.6.2 Firebase Realtime Database

- It is accountable for sending live data from IoT sensors (temperature, gas, sound) directly to the interface of the mobile app.
- It provides immediate alerting and live dashboard refresh.

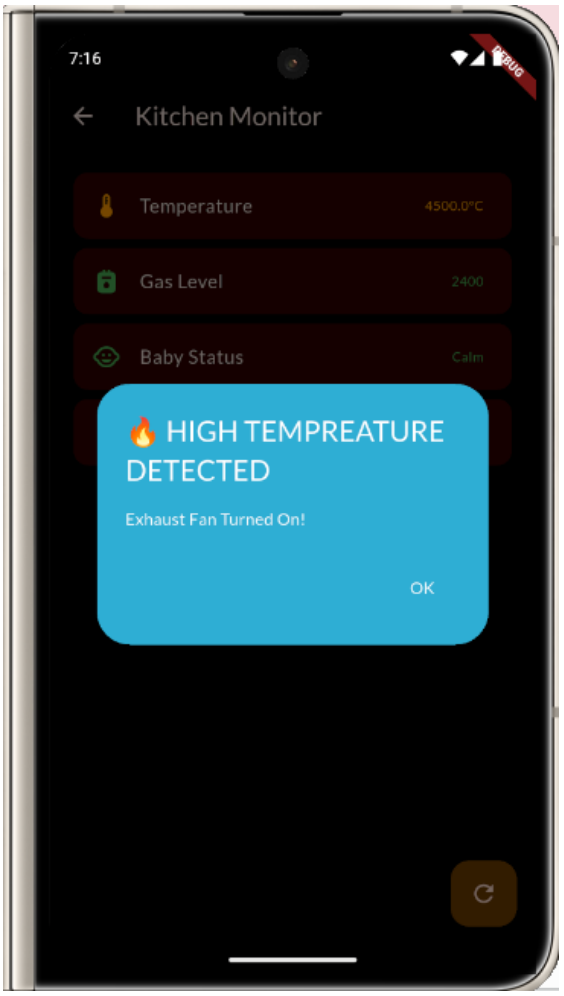

FIGURE 6.13: High Temperature Notification

6.6.3 FAISS Vector Database

- FAISS is a vector-based database employed in the RAG chatbot system. It holds embedded recipe information, allowing semantic search through matching user inputs with the most pertinent content for precise and context-sensitive answers.

6.6.4 Local Caching

- The application locally caches commonly accessed information—such as recipes and chatbot history—on the local device. This enhances performance, lowers network demand, and allows for limited offline capabilities

6.7 System Testing and Validation

In order to provide the Smart Kitchen Assistant application, there was extensive testing strategy used. This involved unit testing, integration testing, and user

testing, all geared towards testing various aspects of the system.

6.7.1 Unit Testing

- Unit tests were performed on individual Flutter UI widgets and backend functions to check if they were correct in isolation. Each module—like the recipe search, chat interface, and alert notifications—was tested for expected functionality.
- Edge cases and exception handling were addressed to provide robustness. This level of testing identified and addressed problems early in the development process.

6.7.2 Integration Testing

Integration testing was conducted to ensure the interaction between important subsystems:

- ESP32 and Firebase Realtime Database: Verified that sensor readings were being sent properly and updating live on the app dashboard.
- Flutter App and Firestore: Tested efficient retrieval of user profiles, recipes, and alerts.
- RAG Chatbot and FAISS: Tested correct retrieval of applicable recipes using semantic search and proper response generation by the language model. This testing assured that all modules were integrated smoothly, enabling the desired use cases.

This phase confirmed that all modules worked together seamlessly, supporting the intended use cases.

6.7.3 User Testing

- User testing consisted of real-world trials conducted on a sample set of target users, such as parents and home cooks, and kitchen users. They were instructed to apply the app to everyday activities like browsing recipes, conversing with the assistant, and reacting to real-time signals from sensor input.

Feedback Highlights:

- Chatbot Usefulness: Users valued the AI chatbot extremely useful for promptly finding recipes using accessible ingredients, particularly during peak cooking times.

- Safety Features: The combination of gas, heat, and audio alerts was seen as a smart safety measure, especially among parents with babies or young children at home.
- The interface was found to be intuitive and easy to use, with praise for the filter functions and instant alerts.

The testing and validation process did verify that the system did achieve its design objectives for performance, safety, and user support.

Chapter 7

Results and Conclusion

7.1 Introduction

The Smart Kitchen Assistant application evaluation yielded informative data about it performance, user use, and real-world benefit. This section provides the key results, identifies meaningful outcomes, investigates implications for real-world kitchen safety and user convenience, and suggests directions for future improvements to bring system's capabilities up to the next level.

7.2 Evaluation Overview

The system was tested both through technical performance evaluation and user feedback analysis. Evaluation parameters consisted of system responsiveness, safety monitoring accuracy, AI chatbot functionality, and the general user experience. The Smart Kitchen Assistant was found to be effective in improving kitchen safety and meal planning via AI integration and IoT-based real-time data monitoring. Major findings are:

- **AI-Assisted Cooking Help:** The RAG-based AI chatbot, integrated with DuckDuckGo and LangChain, offered relevant, contextual guidance to users' cooking-related questions, enhancing decision-making in the kitchen.
- **Kitchen Safety Monitoring:** The system effectively detected possible kitchen hazards like gas leaks, unexpected temperature surges, and baby noise detection. These notifications were presented in real-time, enabling user to respond rapidly to avoid accidents.

- **User Satisfaction:** User comments reported high endorsement of the application's ease of use, interactive AI assistant, and dependable alert system. The majority of users reported greater confidence and ease of use in kitchen application.

- **System Accessibility:** With a Flutter-based mobile interface and Firebase backend, users would be able to access the platform at any time, supporting an easier and more contemporary smart home experience.

7.3 Technical Results

The Smart Kitchen Assistant application showed strong performance in the areas of data management, AI query processing, and integration of hardware and software. Some of the key technical results are:

- **Performance Metrics:** cal results are: • Performance Metrics: The system ensured low-latency performance for sensor alerts and chatbot interactions. Real-time updates were enabled through Firebase streams and event listeners, supporting real-time data delivery to users.

- **Hardware Integration Reliability:** ESP32-based sensor modules provided stable data transmission via Wi-Fi. Failover logic and retry mechanisms were used to manage network unreliability, leading to reliable session-to-session communication.

- **Data Accuracy:** Temperature variations, gas concentration, and audio signals were filtered and processed before alarms were triggered. The system provided high accuracy in hazard detection with minimal false alarms and actionable warnings.

- **Security:** Secure authentication using Firebase guaranteed integrity of user data. Sensor data was encrypted in transit to avoid manipulation or eavesdropping, while Firebase rules guarded against unauthorized access.

- **Scalability:** Modular design accommodates more sensor types and user accounts without loss of system performance. Such flexibility guarantees long-term scalability as more features or users come on board.

7.4 User Experience and Feedback

A number of test users such as students and home chefs used the Smart Kitchen Assistant to test usability and functionality. The results indicated a very high degree of satisfaction and trust in the capabilities of the system.

- **Usability:** The Flutter interface of the app was applauded for its simplicity and ease of navigation. Access to meal categorization and chatbot features was logically arranged, making it easy for users of all ages.
- **Engagement:** Dynamic responses from the AI chatbot and real-time notifications kept the users engaged. Users showed interest in more personalized recommendations based on their historical interactions.
- **Feedback:** User interviews and in-app surveys gave constructive recommendations, including the addition of voice interaction and finer sensor readings, which are currently under consideration for an upcoming update.

7.5 Future Enhancements

Based on the user testing and the technical experience, the following future additions are proposed to enhance and extend the Smart Kitchen Assistant:

- **Voice Command Integration:** Support for voice interactions will further the hands-free convenience, particularly when the user is cooking or working with utensils.
- **Enhanced Sensor Suite:** Adding smoke, motion, and humidity detectors will create a better kitchen safety monitoring system.
- **Personalized Meal Recommendations:**By monitoring user preferences and previous interactions, the AI chatbot can offer more personalized meal planning and cooking guidance.
- **Home Automation Integration:** Adding the ability to support integration with smarter kitchen appliances (smart ovens, exhaust fans) may enable automated sensor alert responses, e.g., lighting ventilation during high gas levels.
- **Augmented Reality Recipes:** Using AR-based recipe guides can provide users a step-by-step, interactive cooking tutorial, enhancing the learning and process.

7.6 Conclusion

The Smart Kitchen Assistant app was able to demonstrate effectively how it transform contemporary kitchen settings by integrating AI, IoT, and mobile . It offered real-world solutions for real-time hazard monitoring, cooking help, and an efficient user interface. The modular and scalable nature of the system guarantees the ability to accommodate future needs. Though the app has been effective in ensuring kitchen safety and convenience, there are still chances of incorporating more sophisticated technologies and features. Through the integration of user-suggested improvements and a further deployment of AI and IoT capability, the Smart Kitchen Assistant can keep evolving into a comprehensive intelligent kitchen environment. The experience gained from this project helps to drive the development of smart home technology and can be used to guide future

References

[1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," 2021.

[2] L. M and J. J. Jeya Sheela M.E, "Designing an iot based kitchen monitoring and automation system for gas and fire detection," in *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*, 2022, pp. 346–353.

[3] N. Kanataria, K. P. Patel, H. N. Patel, P. Goel, K. Patel, and D. Garg, "Rag-enhanced large language model for intelligent assistance from web-scraped data," in *2024 9th International Conference on Communication and Electronics Systems (ICCES)*, 2024, pp. 1043–1048.

[4] H.-C. Lee, K. Hung, G. M.-T. Man, R. Ho, and M. Leung, "Development of an rag-based llm chatbot for enhancing technical support service," in *TENCON 2024 - 2024 IEEE Region 10 Conference (TENCON)*, 2024, pp. 1080–1083.

[5] Amaresh, J. Sekar, G. Tiwari, and P. Gupta, "Pdf to ai chat bot using python langchain," in *2024 1st International Conference on Advances in Computing, Communication and Networking (ICAC2N)*, 2024, pp. 482–485.

[6] A. Petrillo, M. Travaglioni, F. De Felice, R. Cioffi, and G. Piscitelli, "Artificial intelligence and machine learning applications in smart production: Progress, trends, and directions," *Journal of Computational Design and Engineering*, vol. 8, no. 1, pp. 121–132, 2021.

[7] M. A. b. M. Zain, "Automatic kitchen fan," Diploma thesis, Politeknik Sultan Salahuddin Abdul Aziz Shah, Malaysia, 2020.

[8] R. Cioffi, M. Travaglioni, G. Piscitelli, A. Petrillo, and F. De Felice, "Artificial intelligence and machine learning applications in smart production: Progress, trends, and directions," *Sustainability*, vol. 12, no. 2, p. 492, 2020.