A REPORT

ON

Mobile App for Direct Market Access for Farmers

A PROJECT REPORT

Submitted by

Antosh	20211CSG0069
Swamy	20211CSG0046
Shiva Prasad	20211CSG0049

Under the guidance of

Dr. Harish Kumar K S

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

At

GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

PRESIDENCY UNIVERSITY

BENGALURU

MAY 2025

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project report **"Mobile App for Direct Market Access for Farmers"** being submitted by **Swamy B U, Antosh, Shiva Prasad** bearing roll numbers **20211CSG0046, 20211CSG0069, 20211CSG0049** in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a Bonafide work carried out under my supervision.

Dr. Harish Kumar K S	**Dr. Saira Banu Atham**
Assistant Professor	Professor & HOD
Presidency School of CSE	Presidency School of CSE
Presidency University	Presidency University

Dr. MYDHILI NAIR	**Dr. MD. SAMEERUDDIN KHAN**
Associate Dean	Pro-Vc Presidency School of Engineering
Presidency School of CSE	Dean – Presidency School of CSE&IS
Presidency University	Presidency university

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **"Mobile App for Direct Market Access for Farmers"** in partial fulfillment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering**, is a record of our own investigations carried under the guidance of **Dr. Harish Kumar K S, Assistant Professor, School of Computer Science and Engineering, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other degree.

Student Name	**Roll NO**	**Signatures**
Shiva Prasad M	**20211CSG0049**	
Swamy B U	**20211CSG0046**	
Antosh	**20211CSG0069**	

ABSTRACT

Businesses that want to enhance users experience and stay ahead in the competition cannot neglect the importance of customer support. One of the issues that traditional support models face is slow response rates which impacts service delivery badly. VirtuServe is an AI-supported software that uses Generative AI, NLP, and Machine Learning to aid in the development of cutting-edge customer service systems that involve cutting down on repetitive tasks, 24/7 customer query resolution, and human-machine fusion. This minimizes operational expenditure while increasing the satisfaction of customers.

Similarly, Campus also discusses the paradigm development and realization of a rule-based chatbot of AI has stressed healthcare-related questions. The bot uses a modular system to be able to deal with billing, diseases, clinical emergencies, and various clinical specialties efficiently. It is built on the React and TypeScript frameworks thus providing a sturdy scalable and maintainable user interface while utilizing various AI models for effective question answering. UIs that are more interactive, better query composer UI that can be expanded, and better type classrooms are some of the key features that make it easily integrative for the healthcare setting.

In this survey, we review the ability of Virtu Serve and its analogues regarding the significant findings and summarizing barriers the research highlights.

ACKNOWLEDGEMENT

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time. We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC, School of Engineering, Presidency School of Computer Science Engineering & Information Science, Presidency University for getting us permission to undergo the project. We express our heartfelt gratitude to our beloved Associate Dean **Dr. Mydhili Nair,** Presidency School of Computer Science Engineering & Information Science, Presidency University, and **Dr. Saira Banu Atham**, Head of the Department, Presidency School of Computer Science & Engineering, Presidency University, for rendering their timely help in completing this project successfully. We are greatly indebted to our guide **Dr. Harish Kumar K S, Assistant Professor** and Reviewer **Mr. Yamunappa**, **Assistant Professor,** Presidency School of Computer Science Engineering, Presidency University for inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work. We would like to convey our gratitude and heartfelt thanks to the CSE7301 University Project Coordinators **Dr. Sampath A K and Mr. Md Zia Ur Rahman,** department Project Coordinator **Dr.HM Manjula** and Git hub coordinator **Mr. Muthuraj.**

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

Name of the Students
Shiva Prasad M
Swamy B U
Antosh

List of Figures

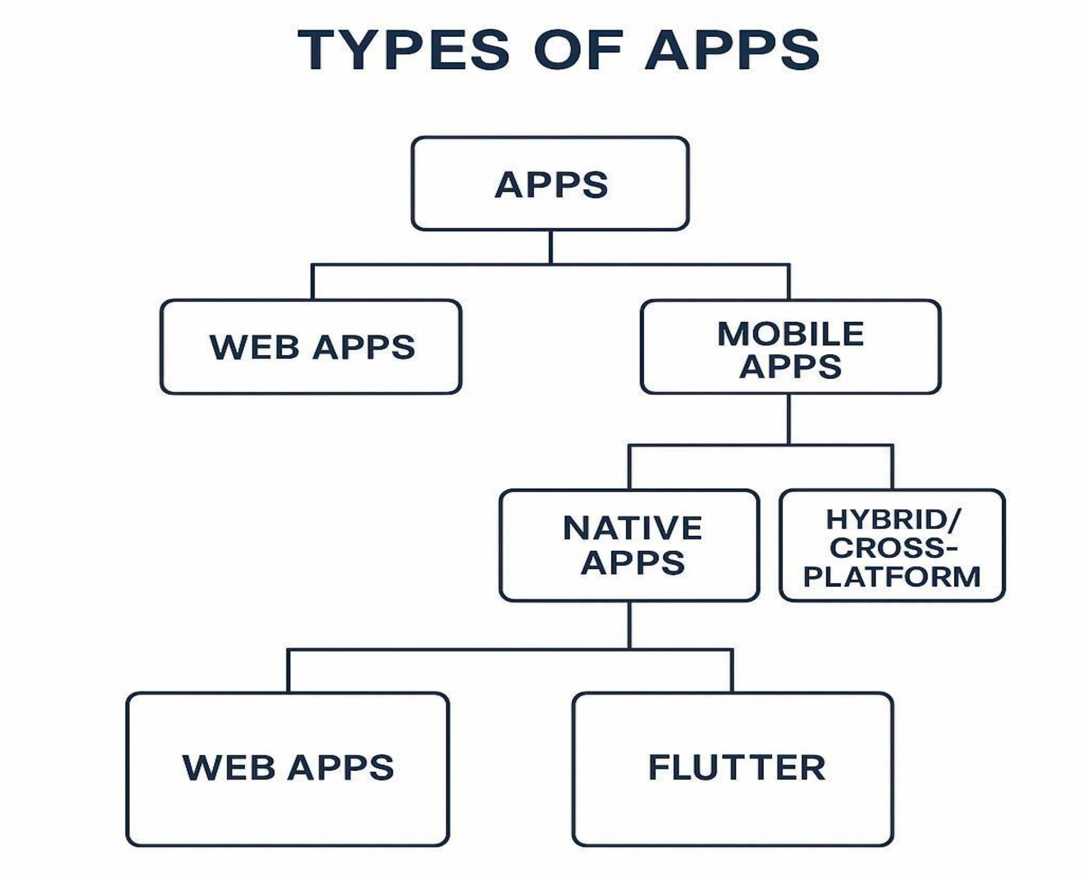
List of Tables

TABLE OF CONTENTS

CHAPTER-1 INTRODUCTION

Especially in underdeveloped nations where a significant portion of the people depends on farming for income, agriculture remains the mainstay of many economies. One of the main problems little and marginal farmers, however, is the lack of direct market access. Historically, a chain of middlemen has dominated agricultural marketing, lowering the farmer's profit margin and openness in pricing.

Mobile technology has become a transforming force in the agriculture industry as smartphones and internet connection proliferate quickly in rural areas. Direct market access via a mobile app lets farmers engage directly with consumers, wholesalers, and stores—so removing intermediaries, lowering post-harvest losses, and guaranteeing



.

Figure 1.1 Graphic representation of the many types of chat bots

Real-time market pricing, demand forecasts, logistical assistance, digital payments, weather updates, and buyer-seller matchmaking can all be provided by such applications. Utilizing digital channels helps the agricultural value chain become more transparent, inclusive, and effective.

1.1 Types of Applications:

Particularly in developing nations, the project "Mobile App for Direct Market Access for Farmers" solves several important problems experienced by the agricultural industry. This initiative is both timely and effective for the following reasons:

1. Empowering Farmers Financially:

Because of many layers of middlemen, farmers sometimes get only a minor part of the ultimate selling price.
Solution: The app guarantees fair pricing and better profit margins by enabling farmers to sell straight to consumers, companies, or merchants.
Farmers in traditional systems only get 25–40% of the ultimate consumer price, according to a Food and Agriculture Organization (FAO) study.

2. Cutting Back on Market Abuse:

Often, middlemen and brokers weigh scales and modify prices.
Real-time market data, digital payments, and rating systems offered by the app foster transparency and accountability.

3. Improving market penetration:

Farmers are often restricted to local mandis (markets), so limiting revenue potential.
A digital market increases regional, national, and international buyers' access.

4. Increasing Food Supply Chain Efficiency:

By combining logistics, the app improves perishable management, delivery tracking, and inventory control—therefore lowering post-harvest losses.

CHAPTER-2
LITERATURE SURVEY

The use of digital technologies in agriculture, particularly by mobile applications, has attracted a lot of interest in the last decade. The review looks at current research in the key areas of relevance to the project, such as market access, agricultural mobile technology, ICT-based interventions, digital supply chains, and available platforms.

2.1 Challenges in Traditional Agricultural Marketing

- Various studies have stressed the inefficiencies of the conventional farm marketing system, particularly in developing nations such as India.
- NABARD (2021) stated that a maximum of 25–35% of the market price at the end is lost to middlemen and post-harvest handling problems.
- Acharya and Agarwal (2010) noted that small and marginal farmers encounter hindrances such as restricted access to mandis, price determination by middlemen, and absence of bargaining capacity.
- The Planning Commission of India observed that poor market infrastructure and lack of transparent pricing mechanisms expose farmers to exploitation.

- These challenges highlight the need for efficient, transparent, and farmer-friendly marketing mechanisms immediately.

2.2 ICT and Mobile Technology in Agriculture

- Information and Communication Technology (ICT), and more specifically mobile phones, have revolutionized agricultural information access for farmers.
- Aker and Mbiti (2010) illustrated how mobile phones lower the cost of information and enhance market coordination in Africa's rural markets.
- Mittal and Mehar (2016) analyzed mobile phone adoption among Indian farmers and established that mobile services enhanced productivity by as much as 25% owing to instant access to market prices and advisory content.

2.3 Digital Platforms and Direct Market Access

- Digital platforms provide an unprecedented opportunity to go around intermediaries and link buyers with farmers directly.
- eNAM (National Agriculture Market) is a government-run program that aggregates APMC mandis online. Yet, the issues of no mobile accessibility, language limitations, and resistance to adoption remain (Reddy et al., 2018).
- Two private digital agribusiness platforms, AgriBazaar and DeHaat, have been successful in buyer-seller matching but are more biased towards B2B commerce and agri-inputs.
- A case study by IFPRI (2019) on "Kalgudi" platform in India revealed a 20–30% income rise for farmers selling directly through the app.
- These studies emphasize the necessity of localized, inclusive, and user-friendly mobile platforms designed to meet farmer requirements.

2.4 Digital Supply Chain and Logistics

- Effective logistics and supply chain integration are key to the success of direct market access.
- Patil and Sherekar (2015) set forth the role of cold chains, real-time monitoring, and digital inventory in minimizing post-harvest losses.
- FAO (2020) indicated that digitization of agriculture may assist in lowering post-harvest losses by as much as 30% if accompanied by logistics management systems.
- Examples such as Ninjacart in India show how networked logistics platforms, enabled by mobile technology, can revolutionize the supply chain for fresh produce.
- Therefore, mobile apps need to incorporate logistics and delivery tracking modules as well to make it complete.

2.5 Rural India's Barriers to Digital Adoption

- Adoption of mobile technology in rural agriculture is not without obstacles despite the promise.
- GSMA (2021) reported that just 32% of rural users in India use smartphones for anything more than basic communication.
- Bhatnagar et al. (2019) identified major challenges as digital illiteracy, insufficient local language content, trust issues with digital platforms, and inadequate connectivity.
- The World Bank (2020) suggested UI/UX design that supports low-literate people and incorporates voice interfaces for greater acceptability.

- This emphasizes the need for inclusive, multilingual, and user-friendly interfaces of mobile apps for farmers.

2.6 Policy Support and Government Initiatives

- The government of India has introduced many schemes that are in line with the objectives of this project:
- Digital India Mission seeks to make the nation a digitally empowered society.
- Doubling Farmers' Income (DFI) Report (2017) highlights market reforms, IT solutions, and improved price realization.
- AgriStack Initiative (2021) is planning to develop a single farmers' database to deliver customized services.
- The project would comfortably fit into these policy guidelines and may also complement existing digital agriculture ecosystems.

2.7 Existing Literature Gaps

- In spite of extensive research, there are some gaps:
- Most current digital platforms serve institutional buyers rather than end consumers.
- Few systems provide end-to-end solutions, covering crop listing, payment, logistics, and support services.
- There is minimal emphasis on local customization, real-time negotiation, and offline support modes for farmers without permanent internet.
- This project seeks to fill these gaps by providing a comprehensive, farmer-centric mobile application.

.

CHAPTER-3
RESEARCH GAPS OF EXISTING METHODS

Even with tremendous advancements in digital platforms for agriculture, there are many limitations and unsolved issues with existing systems. The aim of this project is to bridge those gaps by providing an in-depth, localized, and farmer-centric solution.

3.1: Gap 1: <u>Limited Farmer-to-Consumer (F2C) Connectivity</u>

- Most current platforms (e.g., eNAM, AgriBazaar) concentrate on B2B or Mandi digitization rather than direct farmer-to-consumer models.
- End consumers typically have no recourse to purchase directly from farmers through mobile apps.
- Unavailability of mobile-based F2C marketplaces where farmers can directly connect with local buyers, restaurants, or retailers without intermediaries.

3.2 Gap 2: <u>Lack of Localization and Language Support</u>

- Most applications are developed using English or Hindi, which prevents non-literate or regional language speakers from accessing them.
- Interfaces tend to be too sophisticated for novice smartphone users.
- Unavailability of multilingual, voice-enabled, and icon-based interfaces for inclusive farmer engagement, particularly in tribal and rural regions.

3.3 Gap 3: <u>No Integrated Logistics and Delivery Solutions</u>

- Platforms tend to offer information and listing services but don't deal with logistics (e.g., transportation, packaging, cold storage).
- Farmers have to make arrangements for delivery themselves, which incurs cost and complexity.

3.4 Gap 4: No Real-Time Price Negotiation or Dynamic Pricing

- Most platforms present static price listings.
- Farmers are not able to negotiate or auction their produce in real-time, restricting price optimization.
- No live bidding, price negotiation, and buyer-seller communication features in current solutions.

3.5 Gap 5: Limited Financial Integration

- Delays in payment and lack of transparency are still prevalent due to cash-based or manual payments.
- Limited integration with UPI, digital wallets, or bank accounts.
- Instant, secure digital payments with complete transaction traceability and credit scoring as an option.

CHAPTER-4

PROPOSED METHODOLOGY

4.1 Requirement Analysis

a. Stakeholder Identification

- Identify key users: farmers, consumers, wholesalers, retailers, and logistics providers.
- Consult with agricultural officers, NGOs, and cooperatives for feedback.

b. Needs Assessment

- Undertake surveys and interviews with local farmers to determine:
- Language preferences
- Mobile literacy levels
- Market access challenges
- Document critical pain points and user stories.

4.2 System Design and Architecture

a. Modular Architecture Design

- Develop the app with a modular architecture with:
- Farmer module
- Buyer module
- Admin panel
- Logistics integration

- Payment gateway

b. Technology Stack

- Frontend: Flutter / React Native for cross-platform support
- Backend: Node.js / Django with RESTful APIs
- Database: Firebase / MongoDB for scalability
- Cloud Hosting: AWS / Google Cloud

4.3 <u>Application Development</u>

a. Farmer Module

- Crop listing with images, quantity, and expected price
- Multilingual voice-assisted input (local language support)
- Price trends and market rate dashboard
- Delivery scheduling

b. Buyer Module

- Browse/search produce
- Place orders and negotiate pricing
- Real-time chat or call with farmers
- Track delivery

- c. Admin and Logistics
- Verify farmer registrations
- Manage payment disputes and delivery issues
- Integrate third-party delivery APIs or rural transporters
- GPS-based delivery tracking

4.4 Payment and Transaction Handling

a. Integration of Digital Payment Gateway

- UPI, wallet, and bank transfer support
- Transaction records for all stakeholders
- Auto-release of payment after delivery confirmation

b. Invoicing and Receipts

- Create digital invoices for buyers and sellers
- Transaction history through dashboard

4.5 Testing and Validation

a. Functional Testing

- Ensure that all features are functioning as designed:
- Listing
- Ordering
- Delivery tracking

CHAPTER-5

OBJECTIVES

- To make farmers list agricultural products directly on the app
- Farmers can input crop details, images, expected price, and quantity.
- To facilitate buyers (individuals/retailers) to view, negotiate, and buy produce
- Integrated functions for direct communication and deal closure.
- To incorporate digital payment systems
- Facilitate secure and on-time payments using UPI, bank transfers, or wallets.
- To enable coordination of logistics

CHAPTER-6
SYSTEM DESIGN & IMPLEMENTATION

6.1 Architecture Design

Type: Single-Page Application (SPA)

Core Technologies:

- React: For building reusable UI components.
- TypeScript: For static typing, ensuring code correctness and maintainability.
- Vite: As the build tool for efficient development and optimized production builds.
- TailwindCSS: For styling with a utility-first approach.

Structure:

- Root Component: The application is mounted onto a single HTML <div> (id="root") using React DOM.
- Component-Based Design: Encapsulation of UI logic and styles into small, reusable React components.
- Responsive Design: Enabled via TailwindCSS utilities and media queries.

Frontend Flow:

1. HTML loads the main.tsx entry point, initializing the React app.
2. React manages routing, rendering, and state updates dynamically on the client side.
3. TailwindCSS provides consistent and responsive styles.

Implementation Details:

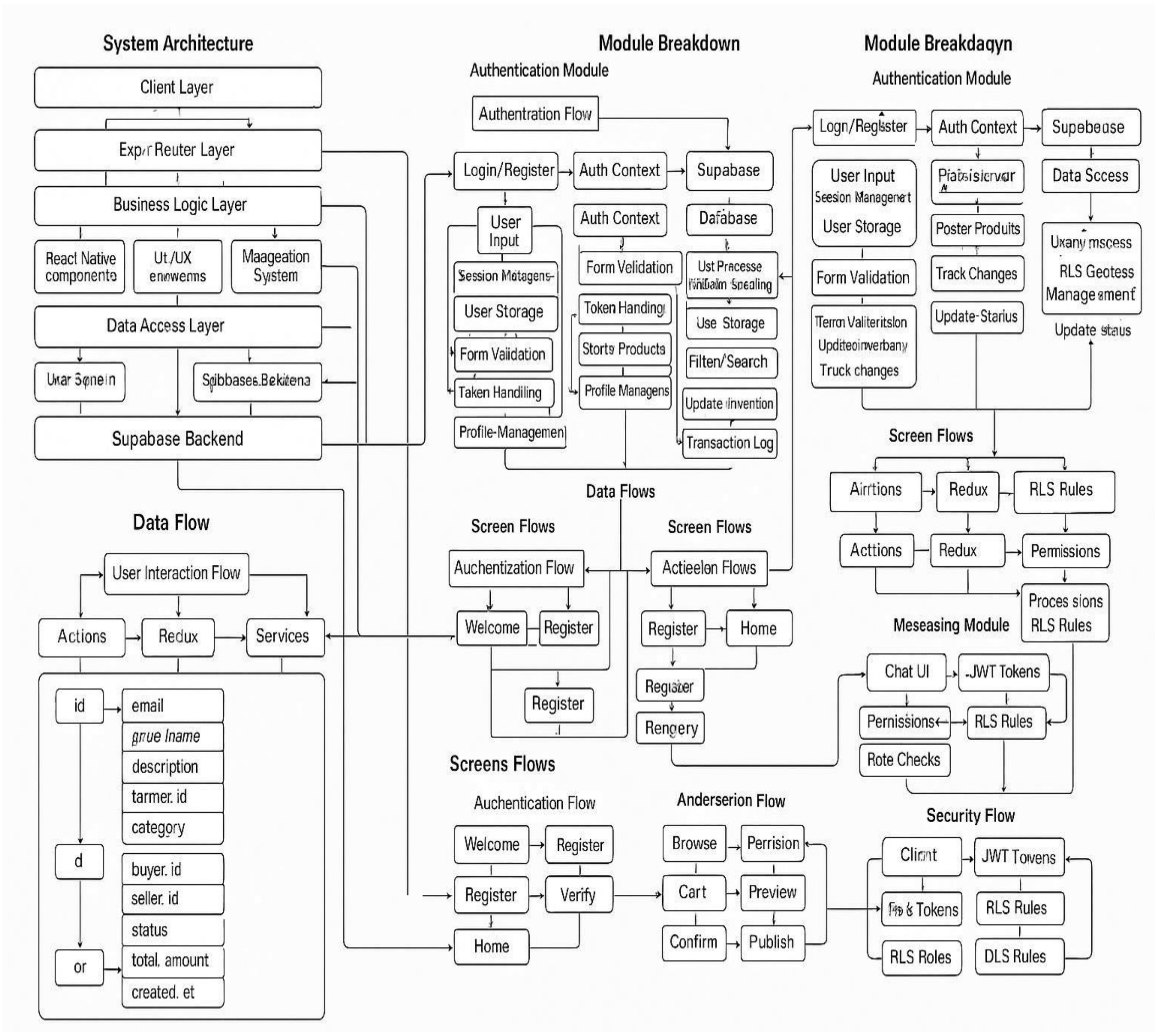


Figure 6.1 Implementation Workflow

1. <u>**Development Framework:**</u>

- Framework: The project uses React with TypeScript for frontend development.

- Key Files: App.tsx, main.tsx, index.html.

- Build Tool: Utilizes Vite, evident from vite.config.ts, offering fast bundling and development server capabilities.

- Styling: Implements Tailwind CSS, as suggested by the presence of tailwind.config.js and postcss.config.js.

2. <u>**Model Integration:**</u>

- Primary File: chatbot.ts in the src/utils directory likely handles AI model interactions.

- <u>Integration Details:</u>
- Communication: Likely uses an API-based mechanism to interact with the AI model.
- Categorization: Modular query handling via queryCategories/, covering diverse topics like billing, emergencies, and diseases.

- Data Management: Designed for scalability, with a focus on modularizing query logic.

3. **Unique Features:**

Custom Query Categories:

- Handles a diverse range of healthcare-related queries, such as diseases, specialties, and billing.
- Extensible architecture allows seamless addition of new query categories.

Dynamic Component Architecture:

- Role-specific UIs, with distinct components for login and chat functionalities.
- Reusable subcomponents, such as LoginHeader.tsx and ChatInput.tsx.
- Maintains a clean separation of concerns, ensuring high maintainability.

Robust Type Safety:

- Strong typing with TypeScript minimizes runtime errors.
- Centralized type definitions in the types directory.
- Facilitates scalability with predictable structures.

6.2 Development Workflow

6.3 Hot Module Replacement (kklld4rst m`ikhg7ty5HMR): Enabled via Vite, allowing real-time updates without a full page reload.

6.4 Linting and Type Checking:

6..1 ESLint ensures adherence to coding standards.

6..2 TypeScript provides type safety during development and builds.

- dev: Starts the development server with live reloading.
- build: Builds the production-ready app.
- lint: Runs ESLint to check for code issues.
- preview: Serves the production build locally for testing.

- TailwindCSS Integration:
  - CSS utilities are applied directly in JSX/TSX files for styling.
  - Custom configurations can be added in the tailwind.config.js file to extend the default theme or add plugins.
- PostCSS and Autoprefixer:
  - Autoprefixer ensures cross-browser compatibility for CSS.
  - PostCSS processes styles efficiently during builds.

Frontend architecture

```
src/
├── components/              # React Components
│   ├── chat/                # Chat-related components
│   │   ├── ChatHeader.tsx     # Header with user info and logout
│   │   ├── ChatInput.tsx      # Message input component
│   │   └── MessageList.tsx    # Chat messages display
│   ├── login/               # Login-related components
│   │   ├── LoginHeader.tsx    # Login page header
│   │   ├── LoginInput.tsx     # Username input
│   │   └── RoleSelector.tsx   # User role selection
│   ├── ChatInterface.tsx      # Main chat interface
│   └── LoginForm.tsx          # Main login form
```

6.3 Chat Response System

```
src/utils/
├── chatbot.ts               # Main chatbot logic
└── queryCategories/           # Organized response categories

├── diseases/                # Disease-specific responses
├── management/                # Healthcare management responses
├── patients/                # Patient-related responses
├── skincare/                # Skincare-specific responses
├── specialties/             # Medical specialties
└── other categories           # Various healthcare topics
```

6.4 Type System

```
src/types/
└── index.ts              # TypeScript interfaces
    ├── Message            # Chat message structure
    ├── User               # User information
    ├── ChatResponse       # Bot response structure
    └── QueryCategory      # Response category structure
```

Core Features

- User Authentication (staff/patient roles)
- Real-time Chat Interface
- Smart Response System
- Role-based Access Control

Technical Stack

- React 18
- TypeScript
- Tailwind CSS
- Vit

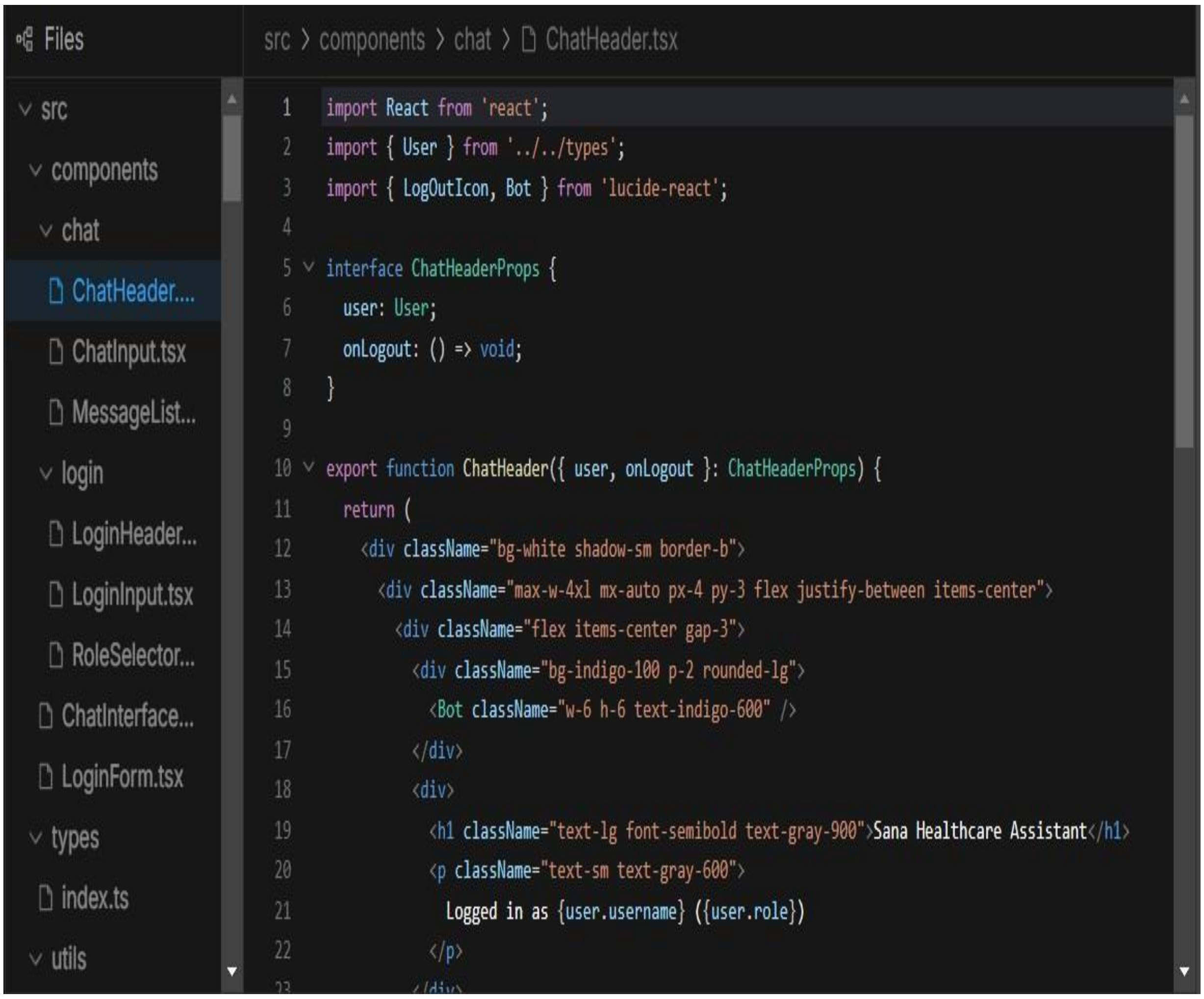
Key Design Patterns

- Component-based Architecture
- Custom Hook Pattern
- Singleton Pattern (for chat service)
- Repository Pattern (for response categories)
- Strategy Pattern (for response generation)

WorkFlow

```
graph TD
    A[User Starts] --> B{Has Account?}
    B -->|Yes| C[Login]
    B -->|No| D[Enter Username]
    D --> E[Select Role]
    E --> C

    C --> F[Chat Interface]

    F --> G[User Types Message]
    G --> H[Message Processing]

    H --> I[Priority Check]
    I -->|Emergency Keywords| J[Emergency Response]
    I -->|Normal| K[Category Matching]

    K --> L[Generate Response]
    J --> L

    L --> M[Display Response]
    M --> N{Continue Chat?}
    N -->|Yes| G
    N -->|No| O[Logout]

    subgraph "Message Processing"
    H --> P[Keyword Extraction]
    P --> Q[Category Selection]
    Q --> R[Response Selection]
    end

    subgraph "Response Categories"
```

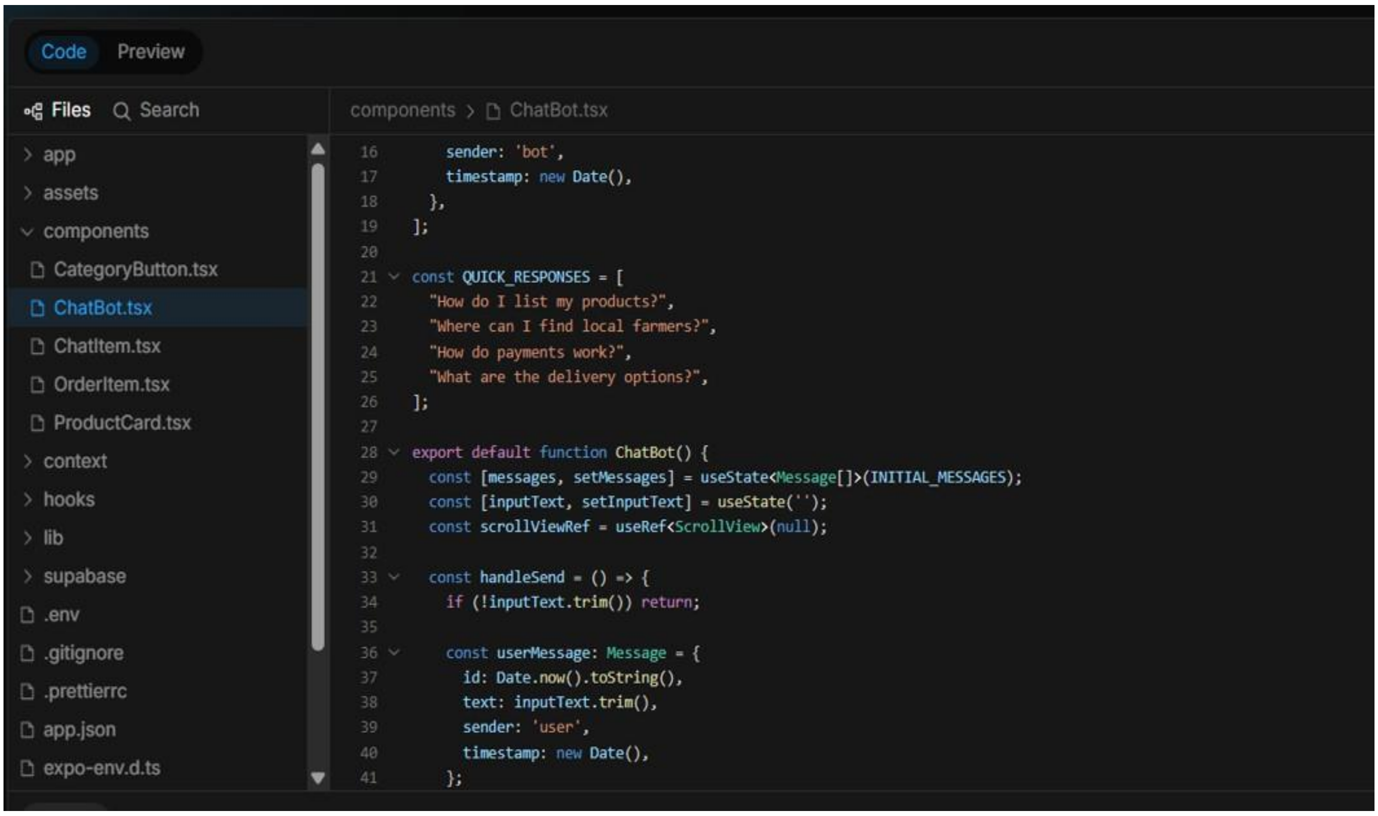
Frontend Components



```
import React from 'react';
import { User } from '../../types';
import { LogOutIcon, Bot } from 'lucide-react';

interface ChatHeaderProps {
  user: User;
  onLogout: () => void;
}

export function ChatHeader({ user, onLogout }: ChatHeaderProps) {
  return (
    <div className="bg-white shadow-sm border-b">
      <div className="max-w-4xl mx-auto px-4 py-3 flex justify-between items-center">
        <div className="flex items-center gap-3">
          <div className="bg-indigo-100 p-2 rounded-lg">
            <Bot className="w-6 h-6 text-indigo-600" />
          </div>
          <div>
            <h1 className="text-lg font-semibold text-gray-900">Sana Healthcare Assistant</h1>
            <p className="text-sm text-gray-600">
              Logged in as {user.username} ({user.role})
            </p>
```

Screenshot 6.1 Frontend Components Code

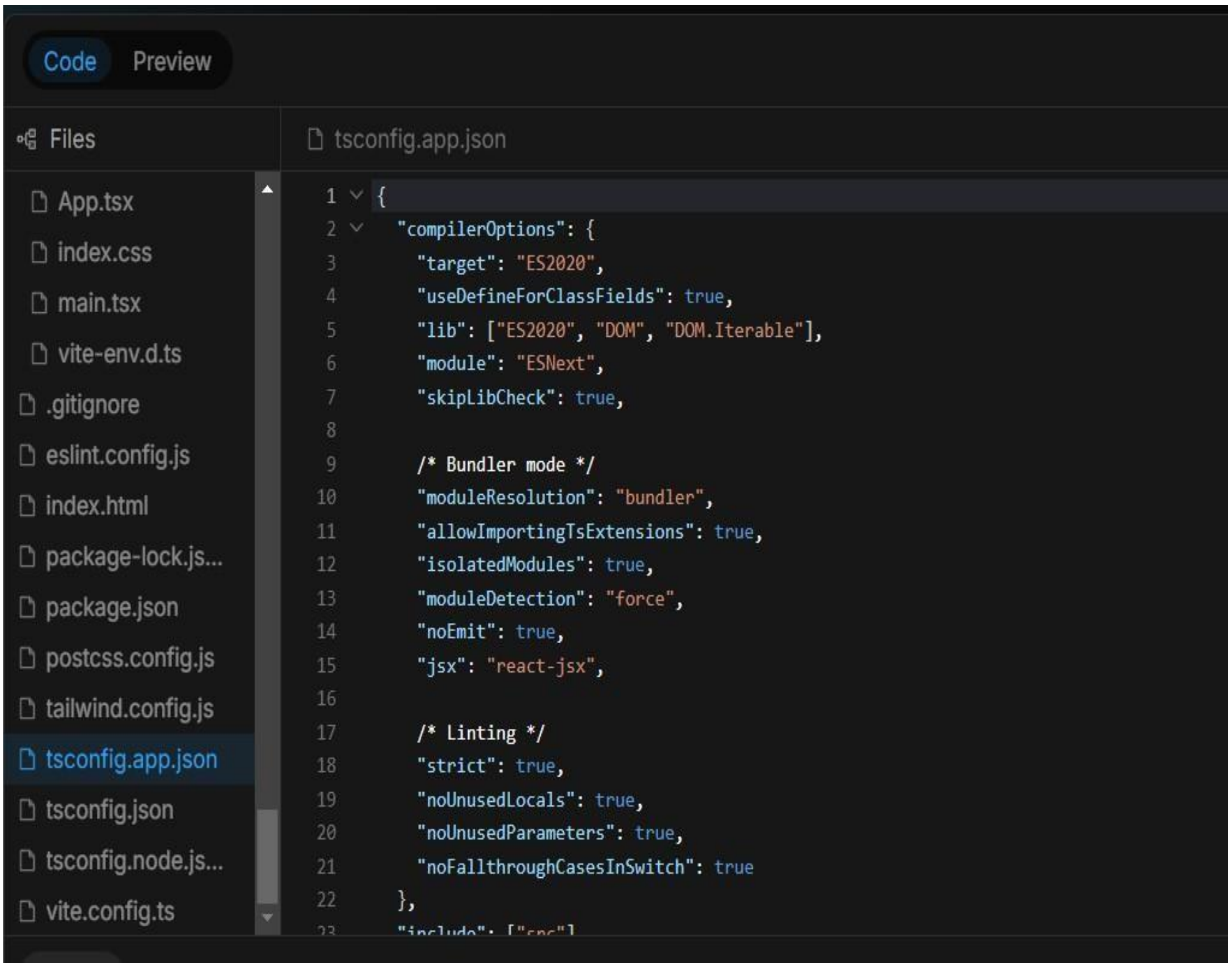
<u>Query Categories</u>



```
      sender: 'bot',
      timestamp: new Date(),
    },
  ];

  const QUICK_RESPONSES = [
    "How do I list my products?",
    "Where can I find local farmers?",
    "How do payments work?",
    "What are the delivery options?",
  ];

  export default function ChatBot() {
    const [messages, setMessages] = useState<Message[]>(INITIAL_MESSAGES);
    const [inputText, setInputText] = useState('');
    const scrollViewRef = useRef<ScrollView>(null);

    const handleSend = () => {
      if (!inputText.trim()) return;

      const userMessage: Message = {
        id: Date.now().toString(),
        text: inputText.trim(),
        sender: 'user',
        timestamp: new Date(),
      };
```

Screenshot 6.2 Query Components

Backend Components



```
{
  "compilerOptions": {
    "target": "ES2020",
    "useDefineForClassFields": true,
    "lib": ["ES2020", "DOM", "DOM.Iterable"],
    "module": "ESNext",
    "skipLibCheck": true,

    /* Bundler mode */
    "moduleResolution": "bundler",
    "allowImportingTsExtensions": true,
    "isolatedModules": true,
    "moduleDetection": "force",
    "noEmit": true,
    "jsx": "react-jsx",

    /* Linting */
    "strict": true,
    "noUnusedLocals": true,
    "noUnusedParameters": true,
    "noFallthroughCasesInSwitch": true
  },
```

Screenshot 6.3 Backend Components Code

Algorithm

In the form of code

// Step 1: Priority Check

```
// Check emergency-related categories first
for (const category of priorityCategories) {
const data = allQueries[category];
  if (data.keywords.some(keyword => lowercaseInput.includes(keyword))) {
    return {
      text:  data.responses[Math.floor(Math.random()  *  data.responses.length)],
      category,
    };
  }
}
```

// Step 2: General Category Check

```
// Check all other categories if no priority match
for (const [category, data] of Object.entries(allQueries)) {
  if (!priorityCategories.includes(category) &&
      data.keywords.some(keyword   =>   lowercaseInput.includes(keyword)))   {
    return {
      text:  data.responses[Math.floor(Math.random()  *  data.responses.length)],
      category,
    };
  }
}
```

```
  // Step 3: Fallback

  return {

    text: "I apologize, but I'm not sure how to help with that specific query. Would you like to speak with a human representative?",


    category: 'unknown',

  };

}
```

Algorithm in the form of Equation

1. **Message Processing Function:** Let M be the input message and R be the response:

   $f(M) \rightarrow R$ where:

   $M \in \text{String}$

   $R \in \{\text{text: String, category: String}\}$

2. **Category Matching Algorithm:** Let C be the set of all categories and K(c) be the keywords for category c:

   $C = \{c_1, c_2, ..., c_n\}$

   $K(c) = \{k_1, k_2, ..., k_m\}$

   $P = \{p_1, p_2, ..., p_i\}$ // Priority categories

   $\forall M \in \text{String}$:

   First check: $\forall p \in P$:

   if $\exists k \in K(p) : k \in M \rightarrow$ return $R(p)$

   Then check: $\forall c \in (C - P)$:

   if $\exists k \in K(c) : k \in M \rightarrow$ return $R(c)$

3. **Response Selection:** For each category c, let R(c) be the set of possible responses:

   $R(c) = \{r_1, r_2, ..., r_k\}$

   Selected response = R(c)[random(0, |R(c)| - 1)]

4. **Message State Management:** Let S be the chat state:

   $S = \{m_1, m_2, ..., m_n\}$ where:

   ```
   m_i = {
   id:        String,
   text: String,
   sender:      {'user'      |      'bot'},
   timestamp: Date
   }
   ```

   New state after message:

   $S' = S \cup \{m_{n+1}\}$

5. **Typing Simulation:** Let t be typing duration:
   t = random(500, 2000) ms
   delay(t) before R

6. **Category Priority Function:** Let w(c) be the weight of category c:

   ```
   w(c) = {
   3 if c ∈ {'emergency', 'ambulance', 'emergencyContacts'}
   1 otherwise
   }
   ```

$\text{Priority}(c_1, c_2) = w(c_1) > w(c_2)$

7. **<u>Matching Score Function</u>:** For input M and keyword k:

match(M, k) = {

1 if k.toLowerCase() $\in$ M.toLowerCase()

0 otherwise

CHAPTER-7

TIMELINE FOR EXECUTION OF PROJECT (GANTT CHART)

Gantt Chart

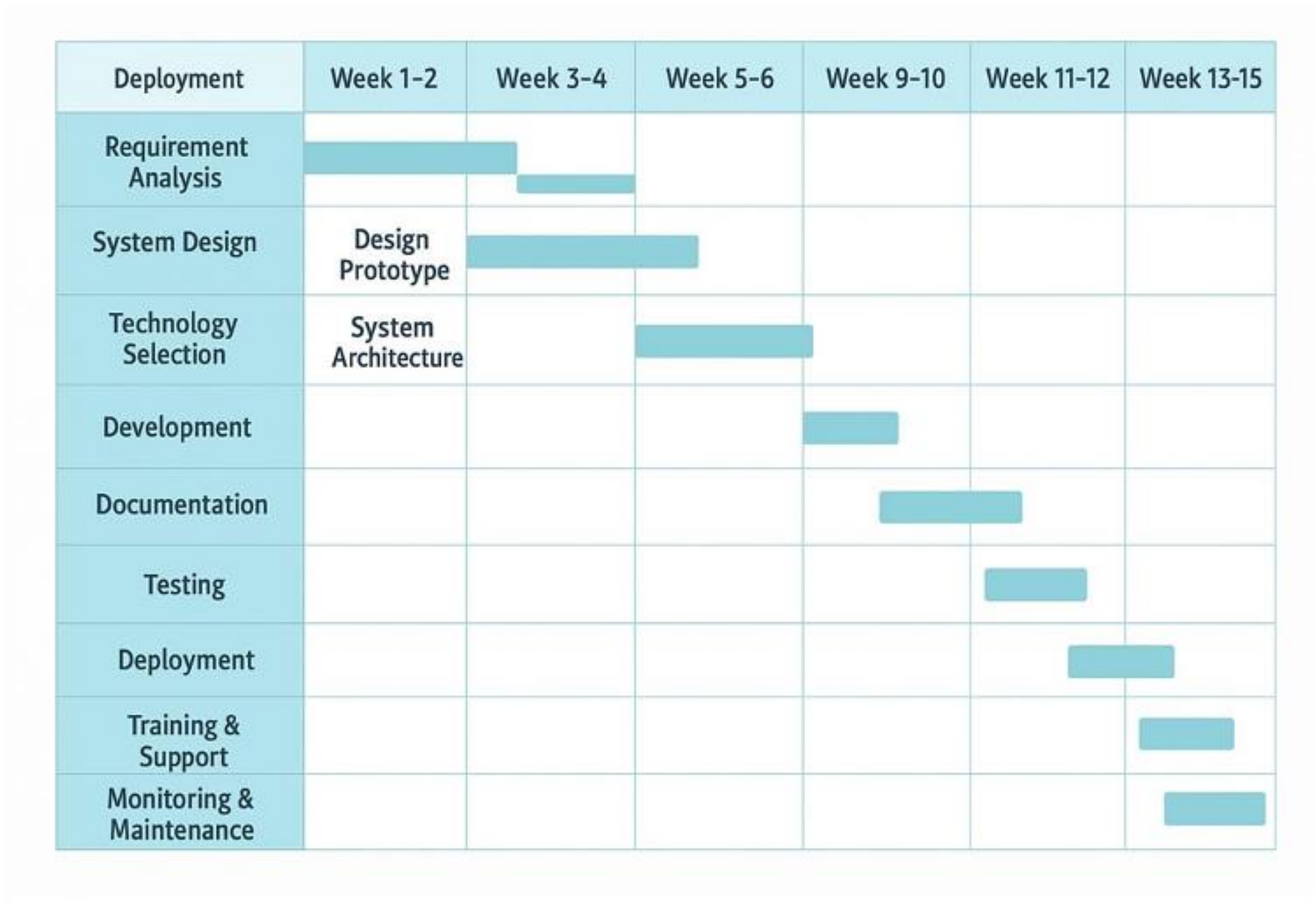


Table 7.1 Timeline of Project

CHAPTER-8
OUTCOMES

8. 1.Functional Outcomes

- Single Page Application (SPA):
  - A fully functional SPA, where navigation between pages/components happens seamlessly without full-page reloads.

- Reusable Components:
  - Modular and reusable React components for building the UI, adhering to React's best practices.

- Responsive Design:
  - A responsive layout achieved using TailwindCSS, ensuring the application works well across devices (mobile, tablet, desktop).

8.2Technical Outcomes

- Optimized Build:
  - A production-ready application with optimized assets (CSS, JS), thanks to Vite's bundling and tree-shaking features.
- Type-Safe Codebase:
  - A TypeScript-enabled codebase to prevent runtime errors and improve maintainability.\
- Linting and Standards:
  - ESLint ensures code adheres to modern JavaScript and React best practices.
  - Strict TypeScript settings enforce high code quality.

- Fast Development Workflow:
    - A development setup with Hot Module Replacement (HMR) for instant feedback during coding.

8.3 Design and User Experience:

- Visually Appealing UI:
    - Tailwind CSS-based UI, customizable with extended themes as needed.
- Interactive Features:
    - Functional UI components, possibly including forms, dynamic buttons, and interactive elements.
- Scalable Architecture:
    - Component-based design makes it easier to add or update features in the future.

CHAPTER-9
RESULTS AND DISCUSSIONS

- **State Management:**

  While the project is functional without global state management, integrating a library like Redux or Zustand could improve scalability for larger applications.

- **Testing Improvements:**

  Basic unit tests were implemented, but further integration and end-to-end testing could improve robustness.

- **API Integration:**

  Currently, the project is static. Adding API integration would enable dynamic data fetching and real-world application scenarios.

CHAPTER-10
CONCLUSION

In conclusion, the integration of modern development tools and libraries such as Vite, React, TypeScript, ESLint, Tailwind CSS, and PostCSS creates an efficient and streamlined environment for building high-quality mobile applications. Vite enhances the development experience with its fast build times and hot module replacement (HMR), significantly speeding up the iterative development process. React provides a powerful and flexible way to create user interfaces with reusable components, while TypeScript introduces static typing, offering better code quality, tooling, and error checking.

ESLint ensures that the code adheres to best practices, improving maintainability and reducing the risk of bugs by enforcing consistent coding standards. Tailwind CSS simplifies styling through utility-first classes, allowing for quick and responsive designs without writing custom CSS. It integrates well with Post CSS for further optimizations like auto prefixing, ensuring broad browser compatibility.

Together, these tools form a cohesive development ecosystem, enabling developers to focus on building robust, scalable applications while ensuring that the codebase remains clean, maintainable, and performant. The combination of high productivity, ease of use, and long-term maintainability makes this tech stack an excellent choice for modern web development projects.

REFERENCES

1. Shekhar, E. S., D. S. Goyal, and U. J. J. A. W. A. L. Jain. "Enhancing customer engagement with AI and ML: Techniques and case studies." *International Journal of Computer Science and Publications, 14 (2), 1-15. IJCSP24B1346. Pdf* (2024).

2. FAZIO, GIUSEPPE. "Meet-AI: heuristics and evaluation platform for AI-infused systems." (2022).

3. Behera, Rajat Kumar, Pradip Kumar Bala, and Arghya Ray. "Cognitive Chatbot for personalised contextual customer service: Behind the scene and beyond the hype." *Information Systems Frontiers* 26, no. 3 (2024): 899-919.

4. Behera, Rajat Kumar, Pradip Kumar Bala, and Arghya Ray. "Cognitive Chatbot for personalised contextual customer service: Behind the scene and beyond the hype." *Information Systems Frontiers* 26, no. 3 (2024): 899-919.

5. Alexander, T. (2024). Proactive customer support: Re-architecting a customer support/ relationship management software system leveraging predictive analysis/AI and machine learning. *Engineering: Open Access*. Retrieved from opastpublishers.com

6. Arya, Stuti, Astha Bhaskar, and Kavya Gupta. "Conversational Ai: A Treatise About Vying Chatbots." In *2024 2nd International Conference on Disruptive Technologies (ICDT)*, pp. 929-934. IEEE, 2024.

7. Gupta, Megha, Venkatasai Dheekonda, and Mohammad Masum. "Genie: Enhancing information management in the restaurant industry through AI-powered chatbot." *International Journal of Information Management Data Insights* 4, no. 2 (2024): 100255.
8. Gupta, M., Dheekonda, V. and Masum, M., 2024. Genie: Enhancing information management in the restaurant industry through AI-powered chatbot. *International Journal of Information Management Data Insights*, *4*(2), p.100255
9. Choudhury, Chinmayee, N. Arul Murugan, and U. Deva Priyakumar. "Structure-based drug repurposing: Traditional and advanced AI/ML-aided methods." *Drug discovery today* 27, no. 7 (2022): 1847-1861.

APPENDIX-A
PSUEDOCODE

#Chat Header

```
import React from 'react';
import {User} from '//types';
import {LogOutIcon, Bot} from 'lucide-react';

interface ChatHeaderProps {
  user: User;
  onLogout: () => void;
}


export function ChatHeader({ user, onLogout }: ChatHeaderProps) {
  return (
    <div className="bg-white shadow-sm border-b">
      <div className="max-w-4xl mx-auto px-4 py-3 flex justify-between items-center">
        <div className="flex items-center gap-3">
          <div className="bg-indigo-100 p-2 rounded-lg">
            <Bot className="w-6 h-6 text-indigo-600" />
          </div>
          <div>
            <h1 className="text-lg font-semibold text-gray-900">Sana Healthcare
Assistant</h1>
            <p className="text-sm text-gray-600">
              Logged in as {user.username} ({user.role})
          </p>

        </div>

      </div>

      <button


        className="flex items-center gap-2 px-4 py-2 text-gray-600 hover:text-gray-900
          hover:bg-gray-100 rounded-lg transition-colors duration-200"
      >

        <LogOutIcon     className="w-5     h-5"     />
        Logout
      </button>

    </div>

  </div>

 );

}
```

#ROLE SELECTOR

```
import React from 'react';


interface RoleSelectorProps {
  role: string;
  onRoleChange: (role: 'staff' | 'patient') => void;
  options: {
    value: 'staff' | 'patient';
    label: string;
    icon: React.ReactNode;
  }[];
}


export function RoleSelector({ role, onRoleChange, options }: RoleSelectorProps) { return
  (
    <div className="space-y-2">
      <label className="block text-sm font-medium text-gray-700">
        Select your role
      </label>
      <div className="grid grid-cols-2 gap-3">
        {options.map((option) => (
          <button
            key={option.value}
            type="button"
            onClick={() => onRoleChange(option.value)}
            className={`
              flex items-center justify-center gap-2 py-2.5 px-4
              rounded-xl text-sm font-medium
              transition-all duration-200 transform
              ${role === option.value
                ? 'bg-indigo-600 text-white shadow-lg shadow-indigo-600/25 scale-[1.02]'
                : 'bg-white text-gray-700 border border-gray-300 hover:border-indigo-300 hover:bg-indigo-50'}
            `}
          >
            {option.icon}
            {option.label}
          </button>
        ))}
      </div>
    </div>
  );
}
```

#Login Header

```
import React from 'react';
import {Stethoscope} from 'lucide-react';
```

```
export  function  LoginHeader()  {
  return (
    <div className="flex flex-col items-center space-y-4">

      <div className="relative">

        <div className="absolute inset-0 bg-indigo-100 rounded-full blur-lg transform scale-


150 opacity-50"></div>

        <div className="relative bg-gradient-to-br from-indigo-500 to-purple-500 rounded-full
p-4">
          <Stethoscope className="w-8 h-8 text-white" />

        </div>

      </div>



      <div className="text-center">

        <h1 className="text-2xl font-bold text-gray-900">Welcome to Sana</h1>

        <p     className="mt-2     text-gray-600">
          Your AI-powered healthcare assistant
        </p>

      </div>




      <div className="flex items-center gap-4 text-sm text-gray-500">

        <div className="flex items-center gap-1">

          <span  className="w-2  h-2  bg-green-500   rounded-full"></span>
          24/7 Support
        </div>

        <div className="flex items-center gap-1">

          <span className="w-2 h-2 bg-blue-500 rounded-full"></span>
          Instant Response
        </div>
      </div>
    </div>
  );
}
```

#<u>**Login Form**</u>

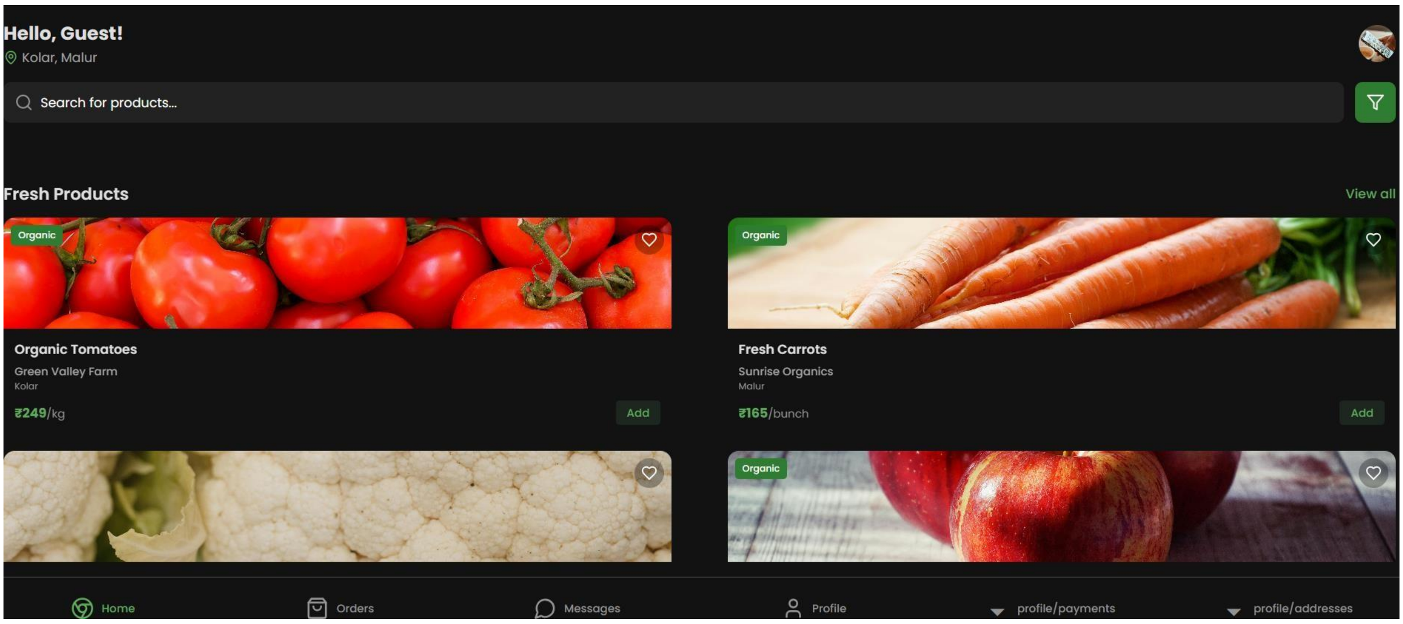
```
import { useState } from 'react';
import { View, Text, TextInput, TouchableOpacity, StyleSheet, Image, ScrollView,
KeyboardAvoidingView, Platform } from 'react-native';
import { useRouter } from 'expo-router';
import { useAuth } from '@/context/AuthContext';
import { Eye, EyeOff } from 'lucide-react-native';

export default function Login() {
const [email, setEmail] = useState('');
const [password, setPassword] = useState('');
const [showPassword, setShowPassword] = useState(false);
const [error, setError] = useState<string | null>(null);
const router = useRouter();
const { signIn } = useAuth();

const handleLogin = () => {
if (!email || !password) {
setError('Please fill in all fields');
return;
}

try {
signIn(email, password);
router.replace('/(tabs)');
} catch (error) {
setError('Invalid email or password');
}
};

return (
<KeyboardAvoidingView
behavior={Platform.OS === 'ios' ? 'padding' : 'height'}
style={styles.container}
>
<ScrollView contentContainerStyle={styles.scrollContent}>
<View style={styles.logoContainer}>
<Image
source={{ uri: 'https://images.pexels.com/photos/2257065/pexels-photo-
2257065.jpeg?auto=compress&cs=tinysrgb&w=1260&h=750&dpr=2' }}
style={styles.logoBackground}
/>
<View style={styles.overlay}>
<Text style={styles.logoText}>FarmConnect</Text>
<Text style={styles.tagline}>Direct from Farm to Table</Text>
</View>
</View>

<View style={styles.formContainer}>
<Text style={styles.title}>Welcome Back</Text>
<Text style={styles.subtitle}>Sign in to continue</Text>

{error && <Text style={styles.errorText}>{error}</Text>}

<View style={styles.inputContainer}>
<Text style={styles.label}>Email</Text>
<TextInput
style={styles.input}
```

```
          placeholder="Enter your email"
          value={email}
          onChangeText={setEmail}
          keyboardType="email-address"
          autoCapitalize="none"
          />
              </View>
```

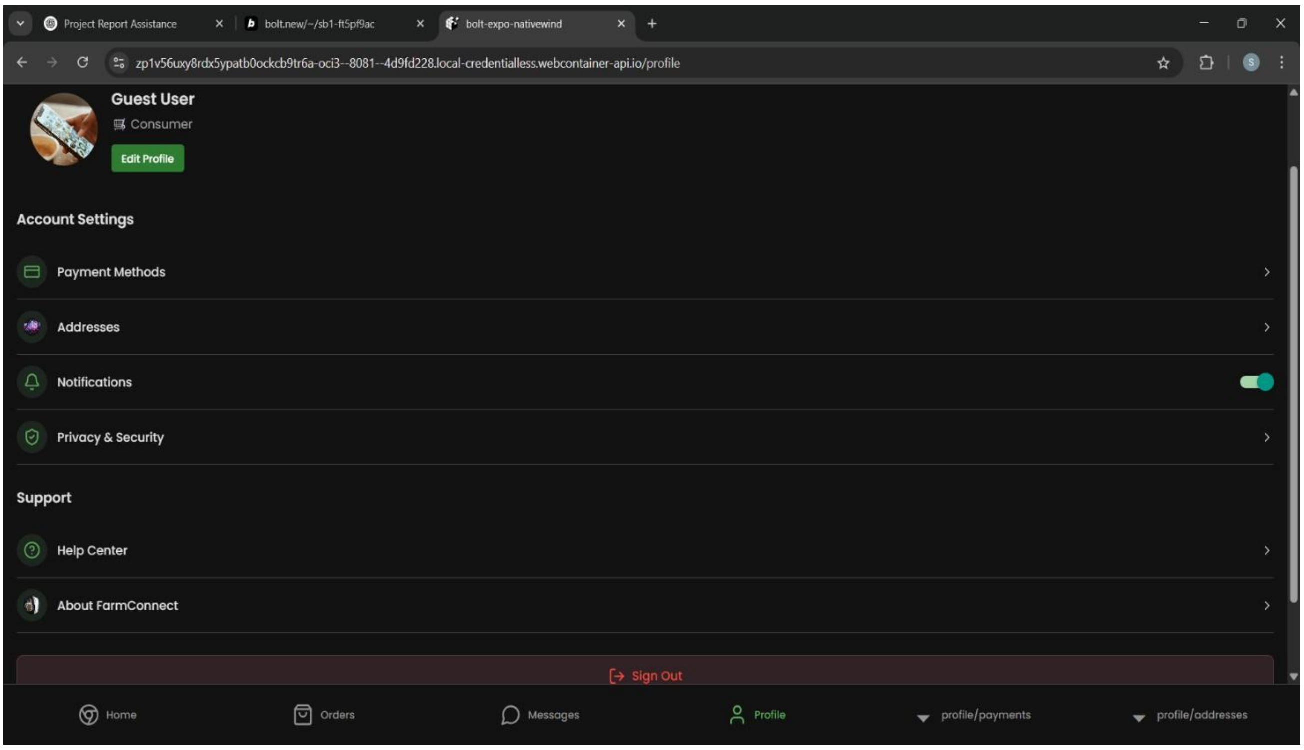
#Payment

```
import { useState } from 'react';
import { View, Text, StyleSheet, TextInput, TouchableOpacity, ScrollView, Alert } from 'react-native';
import { useRouter } from 'expo-router';
import { CreditCard, Plus, Trash2 } from 'lucide-react-native';

interface PaymentMethod {
  id: string;
  cardNumber: string;
  expiryDate: string;
  cardHolder: string;
  type: 'visa' | 'mastercard';
}

export default function PaymentsScreen() {
  const router = useRouter();
  const [showAddCard, setShowAddCard] = useState(false);
  const [cardNumber, setCardNumber] = useState('');
  const [expiryDate, setExpiryDate] = useState('');
  const [cardHolder, setCardHolder] = useState('');
  const [cvv, setCvv] = useState('');
  const [savedCards, setSavedCards] = useState<PaymentMethod[]>([
    {
      id: '1',
    cardNumber: '•••• •••• •••• 4242',
    expiryDate: '12/24',
    cardHolder: 'John Doe',
    type: 'visa',
  },
  {
    id: '2',
    cardNumber: '•••• •••• •••• 5555',
    expiryDate: '10/25',
    cardHolder: 'John Doe',
    type: 'mastercard',
  },
]);

const handleAddCard = () => {
  if (!cardNumber || !expiryDate || !cardHolder || !cvv) {
    Alert.alert('Error', 'Please fill in all fields');
    return;
  }

  const newCard: PaymentMethod = {
    id: Date.now().toString(),
    cardNumber: '•••• •••• •••• ' + cardNumber.slice(-4),
    expiryDate,
    cardHolder,
    type: 'visa',
  };

  setSavedCards([...savedCards, newCard]);
  setShowAddCard(false);
  resetForm();
};

const handleDeleteCard = (id: string) => {
  Alert.alert(
    'Delete Card',
    'Are you sure you want to delete this card?',
    [
      {
        text: 'Cancel',
        style: 'cancel',
      },
      {
        text: 'Delete',
        onPress: () => {
          setSavedCards(savedCards.filter(card => card.id !== id));
        },
        style: 'destructive',
      },
    ],
  );
};

const resetForm = () => {
  setCardNumber('');
  setExpiryDate('');
  setCardHolder('');
  setCvv('');
};

return (
  <ScrollView style={styles.container}>
    <View style={styles.header}>
      <Text style={styles.title}>Payment Methods</Text>
```

APPENDIX-B SCREENSHOTS



Screenshot 10.1 Home Page



Screenshot 10.2 Profile Page

Title: Mobile App for Direct Market Access for Farmers

Abstract: This research paper explores the development and implementation of an AI-powered Mobile Helpdesk System designed to assist farmers in accessing information and support efficiently.

The system utilizes artificial intelligence to automate responses, guide users through agricultural procedures, and facilitate tasks such as form submissions, crop market insights, weather updates, and status tracking.

The objective is to enhance the accessibility of agricultural services, reduce repetitive workload, and improve overall efficiency in farming operations. This paper discusses the technological framework, anticipated benefits, challenges, and potential future developments of the AI-powered mobile helpdesk system.

1. Introduction:

Farmers often face challenges related to information accessibility, slow response times, and high workload due to repetitive queries and manual processes. The integration of AI into a mobile helpdesk system presents an opportunity to enhance operational efficiency and streamline service delivery. This research aims to investigate the feasibility, advantages, and implementation strategies of an AI-driven mobile support system for farmers, helping them access real-time agricultural data, government schemes, and expert advice.

2. Objectives of the Mobile Helpdesk System

- To provide a 24/7 AI-powered assistance platform for farmers.
- To automate responses to frequently asked questions (FAQs) related to crops, livestock, and farming techniques.
- To guide users through agricultural procedures, subsidy applications, and registration processes.
- To facilitate tasks such as form submissions, request tracking, and status updates.
- To provide real-time weather forecasts and market price updates for crops.
- To reduce the workload of agricultural support staff by handling repetitive inquiries.
- To enhance efficiency and improve service delivery within agricultural departments.

3. Technological Framework The AI-powered Mobile Helpdesk System is built using a combination of the following technologies:

- Artificial Intelligence (AI): Utilized for natural language processing (NLP) to understand and respond to queries.
- Machine Learning (ML): Enables continuous improvement of responses based on user interactions and farming trends.
- Mobile Application Development: The system is designed for Android and iOS platforms for accessibility.
- Cloud Computing: Provides secure data storage and seamless access to information.
- Chatbot and Voice Assistant: Allows users to interact with the system through text and voice commands.
- Geolocation Services: Helps farmers access location-specific weather reports and market prices.
- Blockchain Integration: Ensures transparency in transactions and access to verified market information.

4. Expected Benefits

- Improved Accessibility: Farmers can obtain information and support anytime, anywhere.
- Time Efficiency: Reduces the time spent on searching for agricultural information and performing repetitive tasks.
- Cost Reduction: Decreases the reliance on human support staff, lowering operational costs.
- Accuracy and Consistency: Ensures accurate and standardized responses to inquiries.
- Enhanced Productivity: Allows farmers to focus on critical agricultural tasks instead of routine administrative work.
- Market Access: Provides real-time crop pricing and trade opportunities, helping farmers get fair prices for their produce.
- Weather Advisory: Helps farmers make informed decisions based on accurate weather forecasts.

5. Challenges and Limitations

- Data Security and Privacy: Ensuring that sensitive agricultural information is securely stored and accessed.
- AI Training and Accuracy: The AI model requires extensive training to provide accurate and reliable responses tailored to regional farming needs.
- User Adoption and Training: Farmers may need training to fully utilize the system’s capabilities, especially those unfamiliar with digital tools.

- Integration with Existing Systems: Compatibility with agricultural IT infrastructure and government databases may present challenges.
- Internet Connectivity: Ensuring reliable access to digital services in rural areas where internet penetration is low.

6. Future Developments

- Multilingual Support: Expanding language options to cater to diverse farmers across different regions.
- Advanced AI Capabilities: Enhancing AI models for better comprehension, personalized advice, and decision-making assistance.
- Integration with Other Agricultural Services: Connecting with other platforms for a more comprehensive support system, including financial aid programs and insurance.
- User Feedback Mechanism: Implementing feedback loops to continuously refine AI responses and improve user experience.
- IoT Integration: Using smart sensors and IoT devices for real-time monitoring of soil health, irrigation needs, and pest control.
- Supply Chain Enhancement: Linking farmers with buyers, reducing intermediaries, and promoting direct market transactions.

Conclusion:

The AI-powered Mobile Helpdesk System presents a transformative approach to improving farming operations by providing instant access to information and support. Through automation and AI-driven interactions, it enhances efficiency, reduces workload, and ensures a seamless user experience for farmers. The system also empowers farmers with real-time insights into market prices, weather conditions, and best farming practices. While challenges exist, strategic implementation and continuous improvements can maximize its potential, making it a valuable tool for agricultural digital transformation initiatives.

By leveraging AI, ML, and cloud computing, this mobile helpdesk system offers a modern solution to longstanding bureaucratic inefficiencies. With proper adoption strategies and continuous system enhancements, this technology has the potential to revolutionize government service delivery, making it more efficient, accessible, and user-friendly.

.

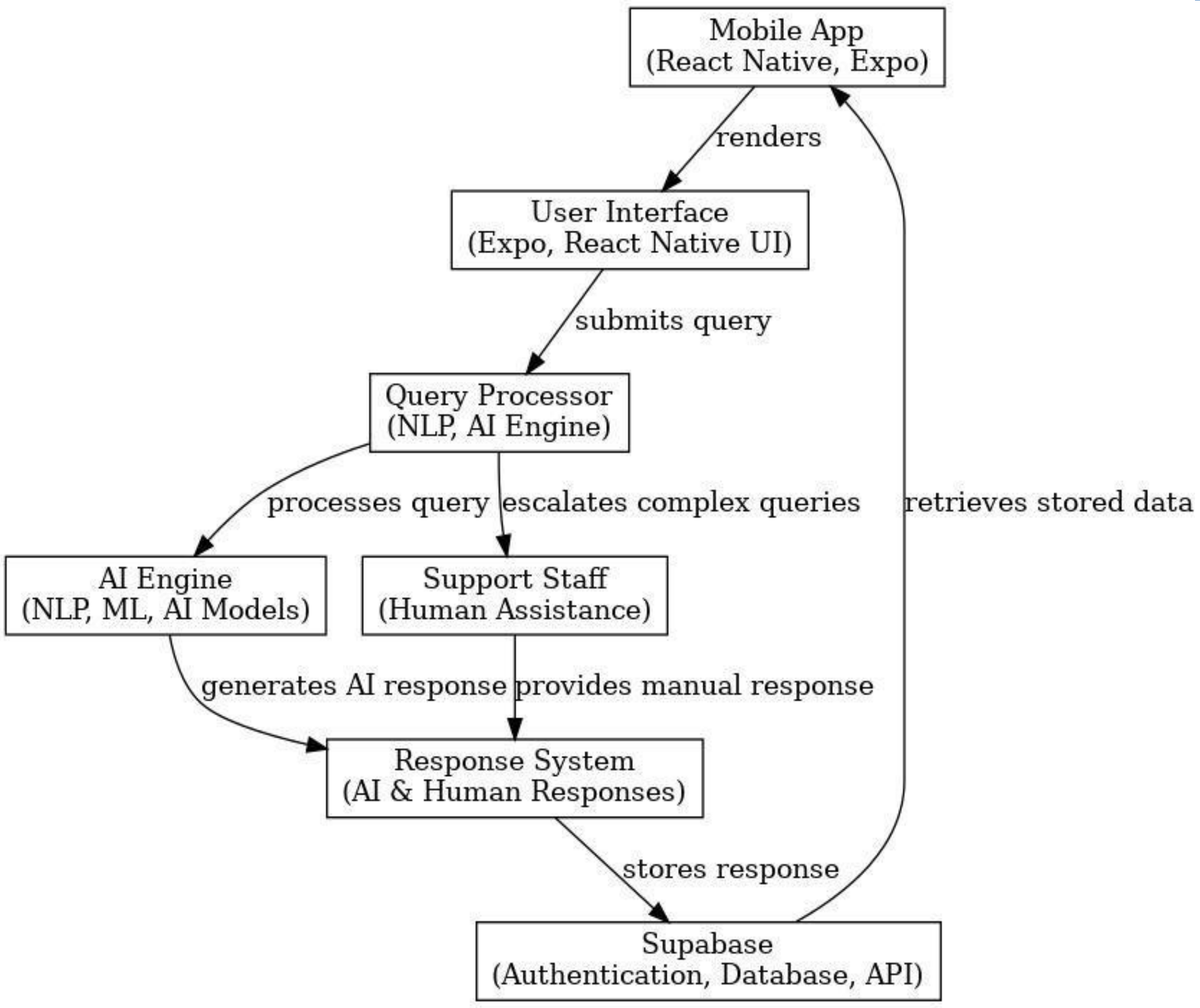

Mobile App (React Native, Expo)
renders
User Interface (Expo, React Native UI)
submits query
Query Processor (NLP, AI Engine)
processes query
escalates complex queries
retrieves stored data
AI Engine (NLP, ML, AI Models)
Support Staff (Human Assistance)
generates AI response
provides manual response
Response System (AI & Human Responses)
stores response
Supabase (Authentication, Database, API)

Figure 10.3 Code Work Flow

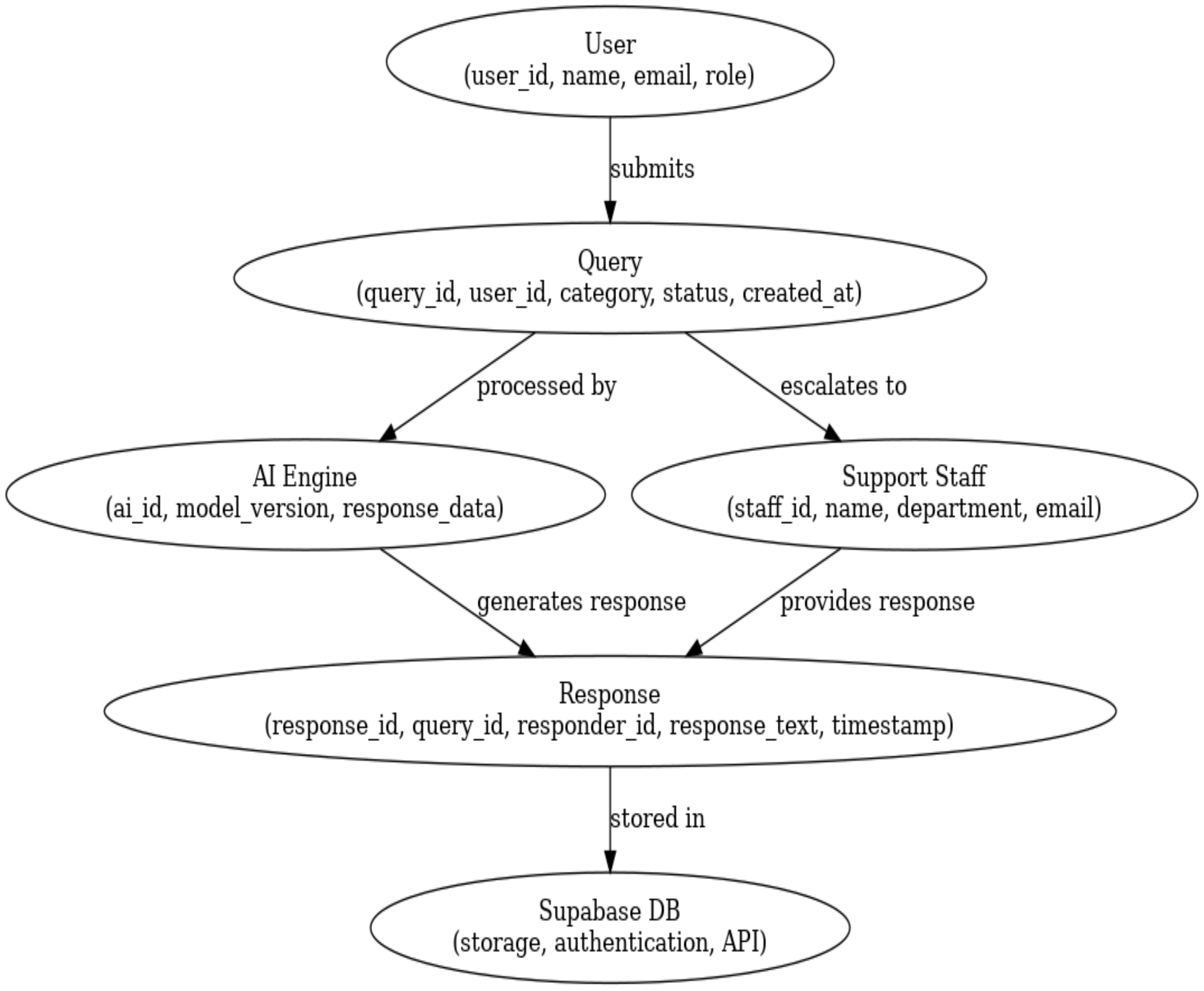

User
(user_id, name, email, role)
submits
Query
(query_id, user_id, category, status, created_at)
processed by
escalates to
AI Engine
(ai_id, model_version, response_data)
Support Staff
(staff_id, name, department, email)
generates response
provides response
Response
(response_id, query_id, responder_id, response_text, timestamp)
stored in
Supabase DB
(storage, authentication, API)

Figure 10.4 User Interaction

Time Complexity: O(n) for searches, O(1) for direct lookups

Space Complexity: O(n) where n is the number of records

Atomicity: Guaranteed for database operations

Consistency: Maintained through RLS policies

Isolation: Provided by database transactions

Durability: Ensured by Supabase persistence

Row Level Security (RLS) Algorithm

Let R = Set of resources

Let U = Set of users

Let P = Set of permissions

Access Control Function A:

A(u: User, r: Resource, op: Operation) → Boolean

For Products:

A(u, p, READ) = true

A(u, p, WRITE) = u.id = p.farmer_id

For Orders:

A(u, o, READ) = u.id ∈ {o.buyer_id, o.product.farmer_id}

A(u, o, WRITE) = u.id = o.buyer_id ∨ u.id = o.product.farmer_id

For Messages:

A(u, m, READ) = u.id ∈ {m.sender_id, m.receiver_id}

A(u, m, WRITE) = u.id = m.sender_id

Real-time Message System Algorithm:

Let M = Set of messages

Let S = Set of subscribers

Let T = Timestamp

Message Delivery Function D:

```
D(m: Message, t: T) = ∀s ∈ S:
  if (s.id = m.receiver_id ∨ s.id = m.sender_id):
    deliver(m, s)
```

Message State Function MS:

```
MS(user_id) = {m ∈ M |
m.sender_id = user_id ∨
m.receiver_id = user_id
  ordered by m.created_at ASC}
```

Order Processing Algorithm:

```
Let O = Order
Let I = Inventory
Let V = Validation Rules

Order Creation Function C:
C(O) = {
  validate: V(O) → Boolean
  process: O → O' where O' = {
  id: UUID,
    buyer_id: UUID,
    product_id: UUID,
    quantity: ℕ,
    total_price: ℝ+,
    status: {'pending'}
  }
  updateInventory: I → I' where
    I'[product_id] = I[product_id] - O.quantity
}

Constraint: ∀o ∈ O: o.quantity ≤ I[o.product_id]
--------------------------------
```

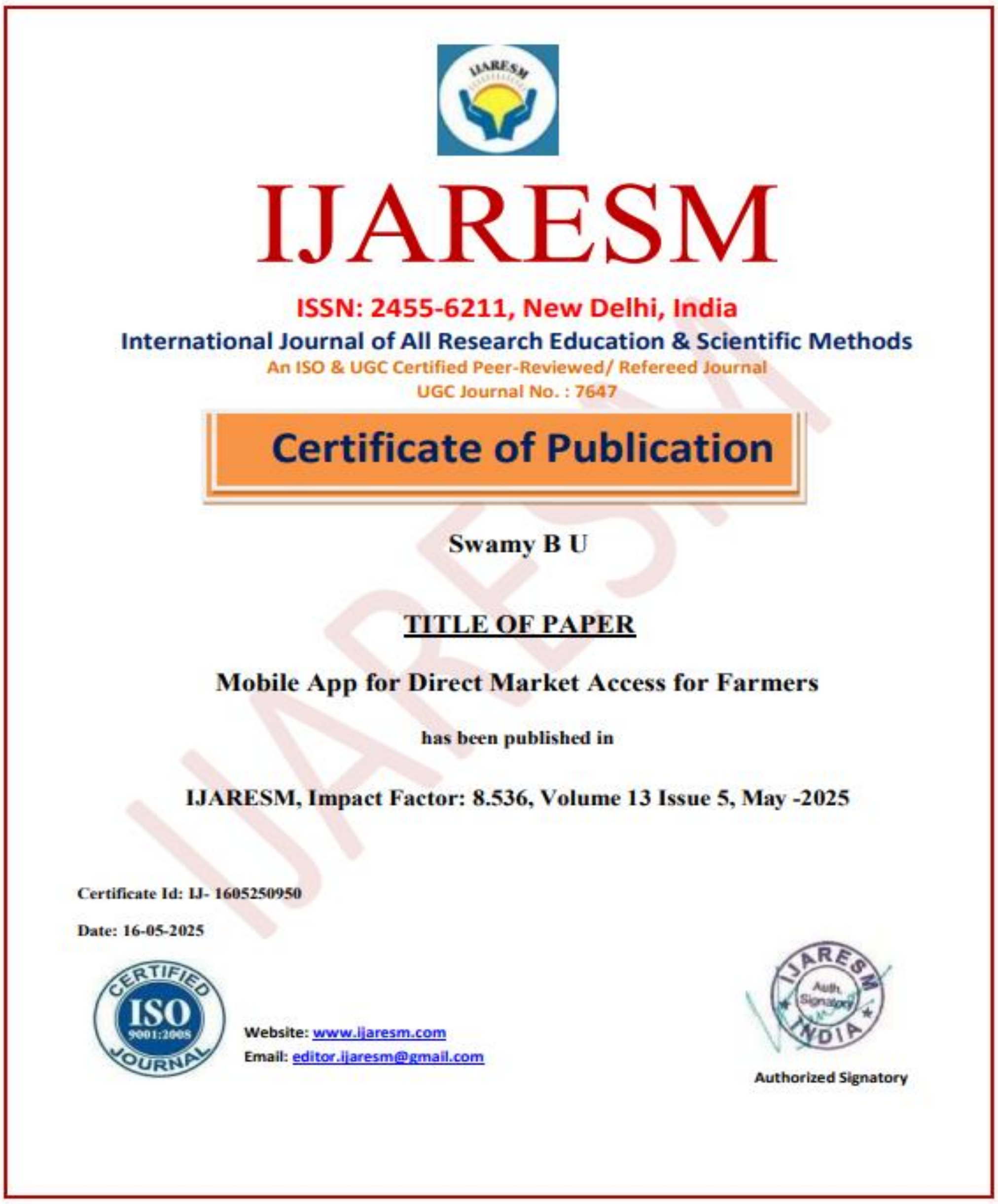
Product Search and Filter Algorithm:

```
Let P = Set of all products
Let Q = Search query string
Let F = Set of filter criteria

Search Function S:
S(P, Q, F) = {p ∈ P | match(p, Q) ∧ filter(p, F)}
```

```
where match(p, Q) = ∀q ∈ tokenize(Q):
  q ⊆ p.name.toLowerCase() ∨
  q ⊆ p.description.toLowerCase() ∨
  q ⊆ p.farmer.toLowerCase()
-----------------------------

Authentication State Management Algorithm:

Let U = User State
Let L = Loading State
Let E = Error State

Function AuthState(t: time):
U(t) = {
  user ∈ Session | null,
  loading ∈ {true, false},
  error ∈ String ∪ {null}
}

State Transition Function T:
T: (U(t), Action) → U(t+1)

where Action ∈ {
  LOGIN(credentials),
  LOGOUT,
  SESSION_CHANGE(session),
  ERROR(message)
}
```

APPENDIX-C
ENCLOSURES

1. Journal publication/Conference Paper Presented Certificates of all students.



IJARESM

ISSN: 2455-6211, New Delhi, India

International Journal of All Research Education & Scientific Methods

An ISO & UGC Certified Peer-Reviewed/ Refereed Journal

UGC Journal No. : 7647

Certificate of Publication

Swamy B U

TITLE OF PAPER

Mobile App for Direct Market Access for Farmers

has been published in

IJARESM, Impact Factor: 8.536, Volume 13 Issue 5, May -2025

Certificate Id: IJ- 1605250950

Date: 16-05-2025

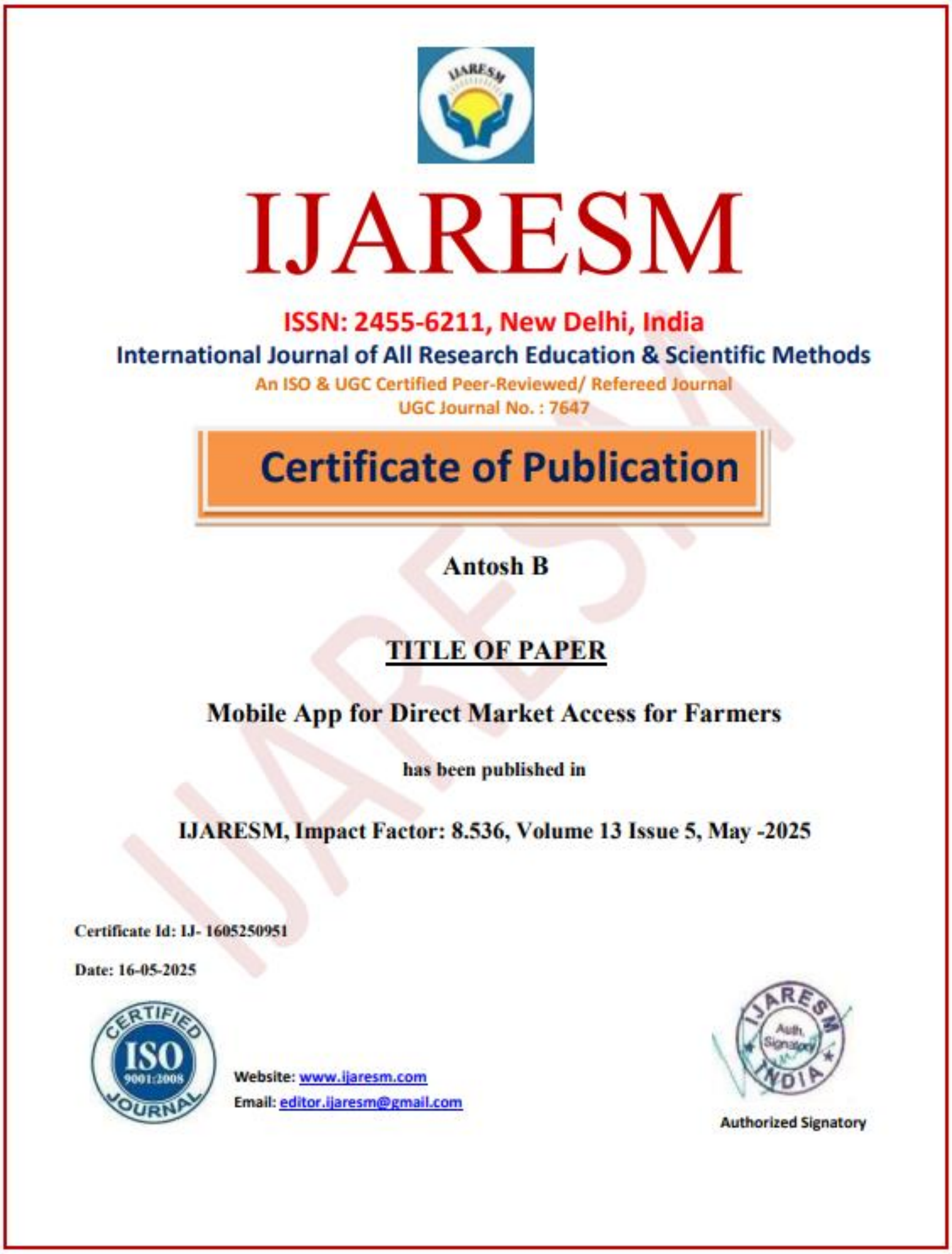
CERTIFIED ISO 9001:2008 JOURNAL

Website: www.ijaresm.com

Email: editor.ijaresm@gmail.com

IJARESM INDIA Auth. Signatory

Authorized Signatory



IJARESM

ISSN: 2455-6211, New Delhi, India

International Journal of All Research Education & Scientific Methods

An ISO & UGC Certified Peer-Reviewed/ Refereed Journal

UGC Journal No. : 7647

Certificate of Publication

Antosh B

TITLE OF PAPER

Mobile App for Direct Market Access for Farmers

has been published in

IJARESM, Impact Factor: 8.536, Volume 13 Issue 5, May -2025

Certificate Id: IJ- 1605250951

Date: 16-05-2025

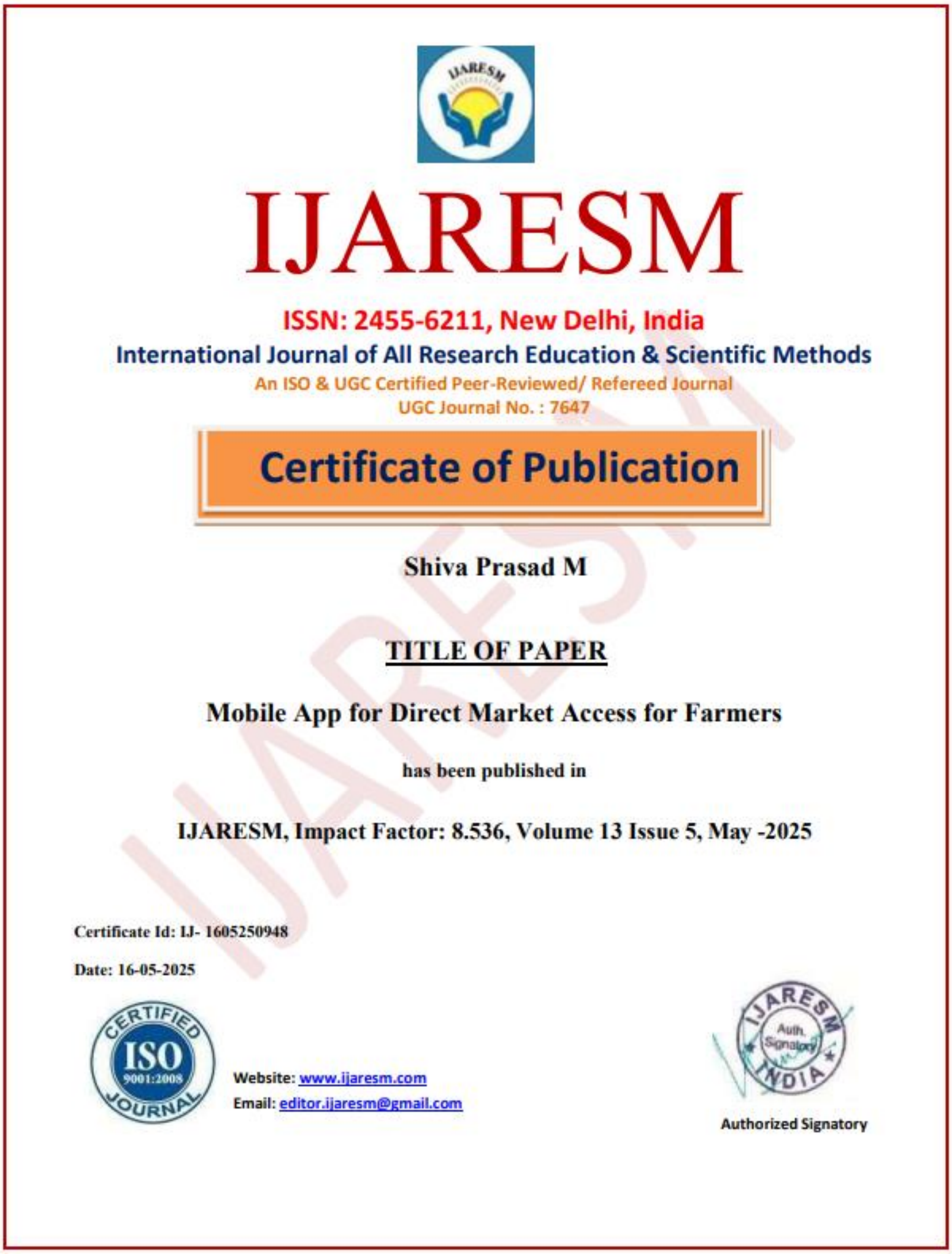
CERTIFIED ISO 9001:2008 JOURNAL

Website: www.ijaresm.com

Email: editor.ijaresm@gmail.com

IJARESM INDIA Auth. Signatory

Authorized Signatory



IJARESM

ISSN: 2455-6211, New Delhi, India

International Journal of All Research Education & Scientific Methods

An ISO & UGC Certified Peer-Reviewed/ Refereed Journal

UGC Journal No. : 7647

Certificate of Publication

Shiva Prasad M

TITLE OF PAPER

Mobile App for Direct Market Access for Farmers

has been published in

IJARESM, Impact Factor: 8.536, Volume 13 Issue 5, May -2025

Certificate Id: IJ- 1605250948

Date: 16-05-2025

CERTIFIED ISO 9001:2008 JOURNAL

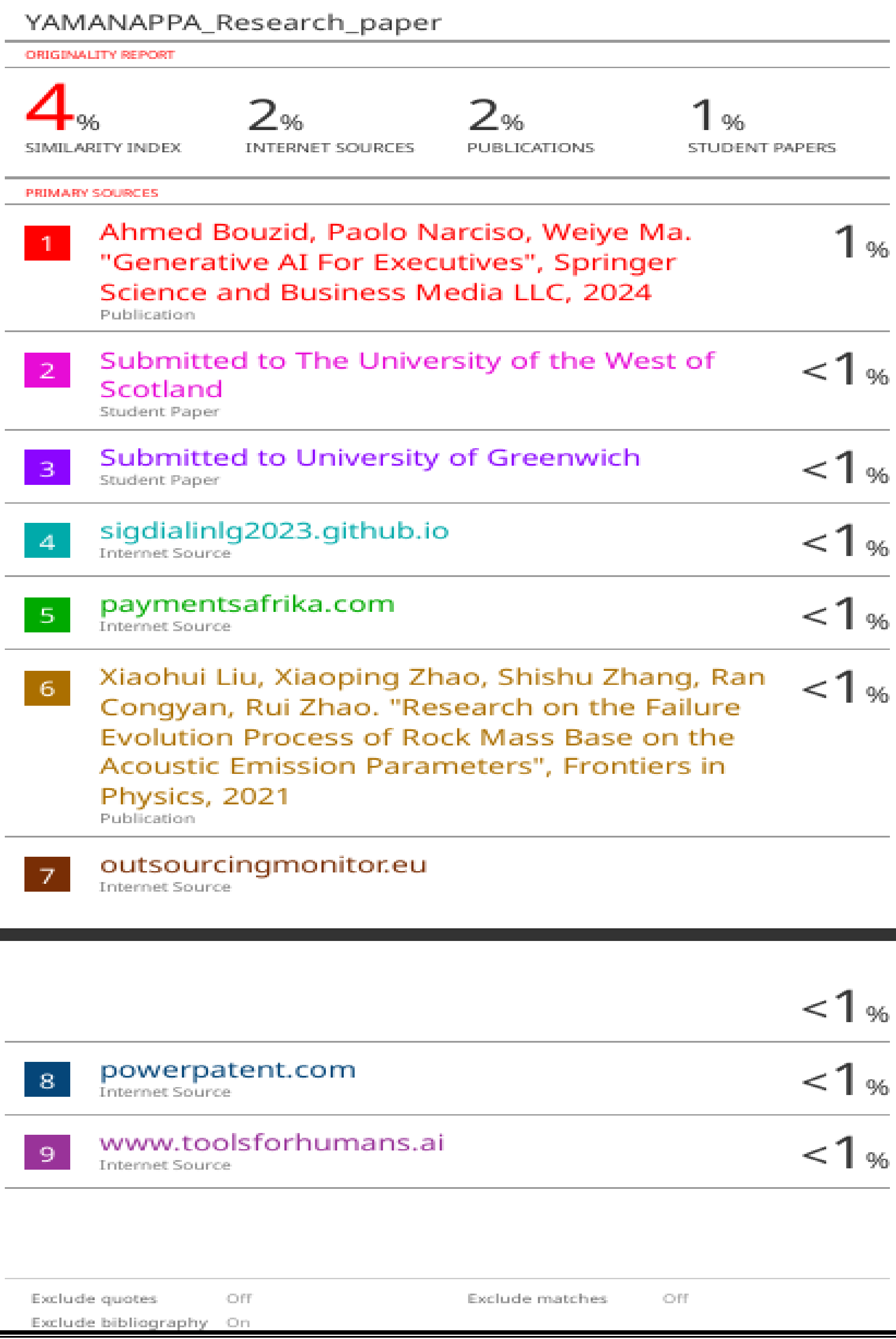
Website: www.ijaresm.com

Email: editor.ijaresm@gmail.com

IJARESM Auth. Signatory INDIA

Authorized Signatory

2. Similarity Index / Plagiarism Check report clearly showing the Percentage (%). No need for a page-wise explanation.



YAMANAPPA_Research_paper

ORIGINALITY REPORT

4%	2%	2%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Ahmed Bouzid, Paolo Narciso, Weiye Ma. "Generative AI For Executives", Springer Science and Business Media LLC, 2024 Publication	1%
2	Submitted to The University of the West of Scotland Student Paper	<1%
3	Submitted to University of Greenwich Student Paper	<1%
4	sigdialinlg2023.github.io Internet Source	<1%
5	paymentsafrika.com Internet Source	<1%
6	Xiaohui Liu, Xiaoping Zhao, Shishu Zhang, Ran Congyan, Rui Zhao. "Research on the Failure Evolution Process of Rock Mass Base on the Acoustic Emission Parameters", Frontiers in Physics, 2021 Publication	<1%
7	outsourcingmonitor.eu Internet Source	<1%
8	powerpatent.com Internet Source	<1%
9	www.toolsforhumans.ai Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		

Sustainable Development Goals

SUSTAINABLE DEVELOPMENT GOALS
17 GOALS TO TRANSFORM OUR WORLD
1 NO POVERTY
2 ZERO HUNGER
3 GOOD HEALTH AND WELL-BEING
4 QUALITY EDUCATION
5 GENDER EQUALITY
6 CLEAN WATER AND SANITATION
7 AFFORDABLE AND CLEAN ENERGY
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
10 REDUCED INEQUALITIES
11 SUSTAINABLE CITIES AND COMMUNITIES
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND
16 PEACE, JUSTICE AND STRONG INSTITUTIONS
17 PARTNERSHIPS FOR THE GOALS
SUSTAINABLE DEVELOPMENT GOALS

1. **App.tsx and main.tsx**:

   - The project is a React-based application with components for user login and a chat interface. This suggests an interactive platform for user communication, possibly in domains like customer service, healthcare, or education.

   .**bolt/prompt and .bolt/config.json:**

   - The project uses a template for a React and TypeScript-based application styled with Tailwind CSS. The focus on "beautiful" and "fully featured" design implies a user-centric approach.

2. **package.json:**

   - The project leverages modern tools and libraries for a responsive and efficient front-end experience, ensuring technological advancement and accessibility.

Mapping to SDGs:

1. SDG 3: Good Health and Well-being:
    - If the chat interface is tailored for healthcare, it could help in telemedicine, patient support, or health education.
2. SDG 4: Quality Education:
    - If the platform facilitates learning through communication, it supports inclusive and equitable education.
3. SDG 9: Industry, Innovation, and Infrastructure:
    - The use of modern frameworks and practices aligns with promoting innovation and building resilient infrastructure.
4. SDG 10: Reduced Inequalities:
    - A focus on user accessibility and inclusivity in design may help reduce inequalities in accessing services or information.