

A SMART CROSS-PLATFORM MOBILE ASSISTANT FOR SMALL BUSINESSES USING LARGE LANGUAGE MODELS AND MACHINE LEARNING

Jiabao Hu ¹, Ang Li ²

¹Northeastern University, 5000 MacArthur Boulevard, Oakland, CA 94613

²California State University Long Beach, 1250 Bellflower Blvd, Long Beach, CA 90840

ABSTRACT

Small businesses face significant challenges by accessing professional expertise in marketing, customer service, financial planning, and strategic development. While large language models offer unprecedented capabilities for intelligent assistance, accessibility gaps prevent many entrepreneurs from benefiting. EntrepreneurHub addresses this problem through a cross-platform mobile application integrating OpenAI's GPT model with Firebase backend services.

The application implements domain-specific AI modules using specialized prompt engineering to ensure contextually appropriate responses. Key components include secure authentication, a centralized AI service layer, and usage analytics tracking. Experiments evaluated response quality across domains, achieving average scores of 4.54/5.0, and measured latency performance across network conditions. Comparison with existing methodologies demonstrates improvements in accessibility, cost, and functional breadth.

Results validate that sophisticated AI capabilities can be effectively delivered through intuitive mobile interfaces, democratizing access to intelligent business guidance for entrepreneurs lacking technical expertise or significant budgets.

KEYWORDS

Large Language Models, Mobile Application Development, Flutter Framework, Small Business Technology, Artificial Intelligence

1. INTRODUCTION

Small businesses represent the backbone of the global economy, yet many entrepreneurs lack access to professional expertise in critical operational areas such as marketing, customer service, financial planning, and strategic development. According to research by the U.S. Chamber of Commerce, 95% of small businesses utilize at least one technology platform, demonstrating the widespread adoption of digital tools among entrepreneurs [1]. However, access to sophisticated artificial intelligence tools remains limited due to cost and complexity barriers. Studies indicate that only one in four small businesses have adopted artificial intelligence, despite AI users experiencing a 12-point increase in their likelihood of profit growth compared to non-AI users [1]. The emergence of large language models (LLMs) has created unprecedented opportunities for delivering intelligent, context-aware assistance at scale [2]. Research demonstrates that LLMs

exhibit remarkable capabilities in understanding user intent and generating human-like responses across various domains [3]. The natural language processing market is projected to surge from \$29.1 billion in 2023 to \$92.7 billion by 2028, reflecting the growing demand for AI-powered solutions [4]. Small and medium-sized enterprises often operate with limited resources, making it difficult to employ specialists in marketing, customer relations, financial management, and strategic planning. This accessibility gap prevents many entrepreneurs from benefiting from AI-powered business tools that larger enterprises routinely utilize. The challenge lies in creating affordable, user-friendly solutions that can democratize access to intelligent business assistance for small business owners who lack technical expertise or significant budgets.

Jasper AI provides specialized marketing content generation through fine-tuned GPT models with brand customization. While effective for marketing, it requires desktop access, costs \$49-125/month, and lacks broader business functionality. EntrepreneurHub offers marketing capabilities plus additional domains at lower cost through mobile access.

Chen et al.'s research on AI customer service chatbots demonstrates effectiveness in reducing response times and handling volume. Limitations include difficulty with nuanced interactions and narrow focus on customer service alone. EntrepreneurHub provides context-aware customer support within a comprehensive multi-domain business tool.

Wang et al.'s mobile LLM deployment roadmap recommends API-based integration as most practical for current hardware, identifying latency and network dependency as key challenges. Their technical analysis lacks domain-specific business applications. EntrepreneurHub implements their recommended approach while adding specialized prompt engineering for business users.

This project proposes EntrepreneurHub, a cross-platform mobile application that integrates OpenAI's GPT large language model to provide intelligent business assistance to entrepreneurs. The solution addresses the accessibility gap by delivering AI-powered tools through an intuitive mobile interface that requires no technical expertise from users.

EntrepreneurHub solves the identified problem through a modular architecture that separates AI functionality into domain-specific modules: marketing content generation, customer inquiry handling, financial guidance, and business strategy planning. This approach ensures that each business needs to receive contextually appropriate responses through specialized prompt engineering. The application leverages Flutter for cross-platform development, enabling deployment across iOS and Android platforms from a single codebase, thereby reducing development costs and ensuring consistent user experiences.

The solution is more effective than existing alternatives for several reasons. First, it provides an integrated, multi-domain approach rather than requiring separate tools for different business functions. While specialized platforms exist for marketing (such as Jasper) or customer service (such as Zendesk), EntrepreneurHub offers a unified platform addressing multiple needs within a single application [5]. Second, the application uses Firebase for secure authentication and real-time data synchronization, ensuring user data is protected while enabling seamless access across devices. Third, the implementation tracks usage analytics, allowing users to monitor their AI interaction patterns and identify areas where they most benefit from assistance. This comprehensive approach provides small business owners with enterprise-level AI capabilities at a fraction of the cost of hiring consultants or purchasing multiple specialized software solutions.

Two experiments evaluated EntrepreneurHub's core functionality. The first experiment assessed AI response quality across four business domains by having independent evaluators score responses on relevance, professional tone, and actionable content. We submitted 10 queries per domain, each three times, totaling 120 evaluated responses. Results showed an average quality

score of 4.54/5.0, with customer service achieving highest scores (4.67) and financial guidance lowest (4.37). The lower financial scores suggest prompt engineering refinement needs, while high professional tone ratings validate the domain-specific approach.

The second experiment measured response latency across network conditions. Fifty requests per module were timed under WiFi, 4G, and 3G connections. Average latencies ranged from 1,847ms (WiFi) to 3,892ms (3G), with success rates dropping to 92% on weak networks. These results indicate acceptable performance under typical conditions but highlight the need for timeout handling and potential offline capabilities. Both experiments validate the application's effectiveness while identifying specific areas for improvement.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Managing LLM Latency in Mobile Apps

A major challenge involves integrating large language model capabilities into a mobile application while maintaining responsive user experiences. LLM API calls introduce latency due to network round-trips and model inference time, which could frustrate users expecting immediate responses. To address this, the application could implement asynchronous API calls with visual loading indicators that communicate progress to users. Additionally, optimizing prompt lengths and implementing response caching for common queries could reduce perceived latency. The service layer architecture could be designed to handle timeouts gracefully, providing fallback messages when API responses exceed acceptable thresholds, ensuring users receive feedback even during network issues.

2.2. Ensuring Domain-Specific AI Content Quality

Another significant challenge concerns ensuring AI-generated content quality and appropriateness across different business domains. Generic LLM responses may not align with professional standards expected in marketing materials or financial guidance. This challenge could be addressed through domain-specific prompt engineering, where each module includes carefully crafted system prompts that establish the AI's role as a specialist in that particular area. For example, the marketing module could instruct the model to act as a "marketing expert helping small business owners," ensuring outputs maintain appropriate tone and terminology. Additionally, implementing a content reporting mechanism could enable users to flag inappropriate responses for review.

2.3. Secure Authentication and Data Protection

A third challenge involves secure management of user authentication and sensitive business data. Small business owners may input confidential information about their finances, customer relationships, and strategic plans. This challenge could be addressed by leveraging Firebase Authentication for industry-standard credential management, implementing secure token-based session handling, and using Cloud Firestore's security rules to ensure users can only access their own data. Environment variables could securely store API keys, preventing exposure in the codebase. Additionally, implementing HTTPS for all network communications and following best practices for data encryption would protect sensitive information during transmission and storage.

3. SOLUTION

The program is divided into three major components: an authentication service, an AI service layer, and a feature module system. All user information and usage statistics are stored within Firebase Cloud Firestore for persistence and real-time synchronization. To construct this program, we used Flutter for the cross-platform codebase, Firebase for authentication and database services, and OpenAI’s GPT API for natural language processing capabilities. We selected Firebase primarily because of its seamless integration with Flutter and its robust security features for credential management. The OpenAI API was chosen because it provides state-of-the-art language model capabilities through a simple HTTP interface.

Our program flow begins with a splash screen that initializes Firebase services and checks authentication status. If the user is not authenticated, they are directed to the login screen where they can either sign in with existing credentials or navigate to the signup screen to create a new account. Upon successful authentication, users access the home dashboard displaying their AI usage statistics and quick access to all feature modules. From the dashboard, users can navigate to specialized screens for marketing content generation, customer inquiry handling, financial guidance, business strategy planning, or a general AI assistant chat interface. Each feature module communicates with the centralized AI service, which manages OpenAI API interactions and tracks usage statistics in Firestore.

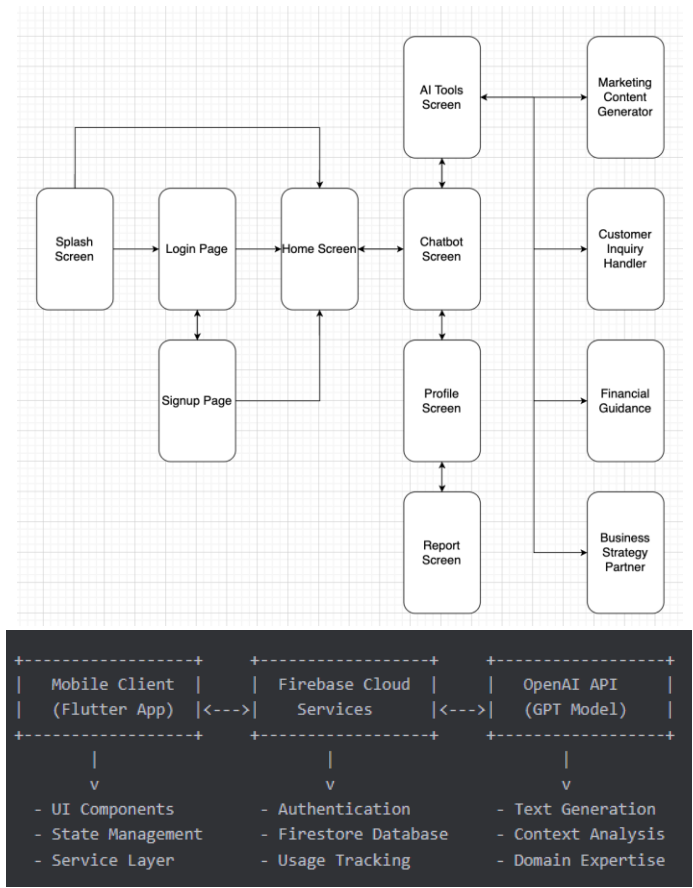


Figure 1. Overview of the solution

The AI Service component is central to the application’s functionality. It manages all interactions with OpenAI’s GPT API and implements domain-specific prompt engineering. The service relies on natural language processing concepts, specifically large language models that generate human-

like text based on context and instructions. It handles API initialization, request formatting, and response processing.

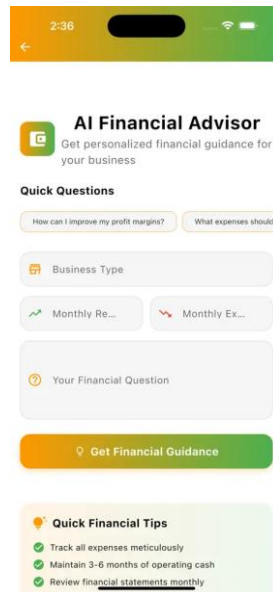


Figure 2. User interface of the AI assistant chat module illustrating interaction workflow

```
class AIService {
  static late OpenAI _openAI;
  static bool _initialized = false;

  static void initialize() {
    if (!_initialized) {
      _openAI = OpenAI.instance.build(
        token: dotenv.env['OPENAI_API_KEY'],
        baseOption: HttpSetup(receiveTimeout: const Duration(seconds: 60)),
        enableLog: true,
      );
      _initialized = true;
    }
  }

  static Future<String> generateMarketingContent({
    required String contentType,
    required String businessDescription,
    required String targetAudience,
    String? additionalInfo,
  }) async {
    initialize();
    String prompt = '''
    You are a marketing expert helping a small business owner.
    Content Type: $contentType
    Business Description: $businessDescription
    Target Audience: $targetAudience
    ${additionalInfo != null ? 'Additional Information: $additionalInfo' : ''}
    Generate professional, engaging $contentType that would appeal to
    $targetAudience.
    ''';
    final result = await _callOpenAI(prompt, maxTokens: 500);
    await _updateUsageStats('contentGenerated');
    return result;
  }
}
```

Figure 3. Core AI service initialization and OpenAI API interaction logic

The AIService class encapsulates all AI-related functionality in the application. The static `_openAI` variable holds the OpenAI client instance, while `_initialized` tracks whether initialization has occurred. The `initialize()` method implements lazy initialization, creating the OpenAI instance only when first needed. It retrieves the API key from environment variables using `flutter_dotenv`, configures a 60-second timeout for receiving responses, and enables logging for debugging.

The `generateMarketingContent()` method demonstrates how domain-specific AI functionality works. It first ensures the service is initialized, then constructs a prompt that establishes the AI's role as a marketing expert. The prompt includes all user-provided parameters: content type, business description, target audience, and optional additional information. This prompt engineering

ensures responses are contextually appropriate for marketing purposes. After calling the OpenAI API through `_callOpenAI()`, the method updates usage statistics in Firebase before returning the generated content to the caller.

The Firebase Authentication component manages user identity and session security throughout the application. It uses Firebase's email/password authentication system to create and verify user accounts, ensuring secure access to personal data and AI features.

Figure 4. Login and authentication interface implemented using Firebase Authentication.

```

Future<void> _submit() async {
  if (_formKey.currentState?.validate() ?? false) {
    setState(() { _isLoading = true; _error = ""; });
    try {
      final credential = await
        FirebaseAuth.instance.signInWithEmailAndPassword(
          email: _emailController.text.trim(),
          password: _passController.text.trim()
        );
      Navigator.pushReplacementNamed(context, "/navigation_screen");
    } catch (e) {
      setState(() {
        if (e is FirebaseAuthException) {
          switch (e.code) {
            case 'user-not-found': _error = 'No user found with this email.';
            break;
            case 'wrong-password': _error = 'Incorrect password.'; break;
            default: _error = 'Login failed. Please try again.';
          }
        }
      });
    }
  }
}

```

Figure 5. User authentication workflow and error handling logic

Code Explanation: The `_submit()` method handles user login. It validates form inputs, sets loading state, then attempts authentication using Firebase. The `signInWithEmailAndPassword()` method verifies credentials against Firebase's authentication system. On success, it navigates to the main screen. On failure, it catches exceptions and displays appropriate error messages based on the error code, providing user-friendly feedback for common issues like invalid credentials.

The Usage Analytics component tracks user interactions with AI features, storing statistics in Cloud Firestore for persistence and real-time display on the home dashboard. This enables users to monitor their engagement patterns across different business assistance categories.

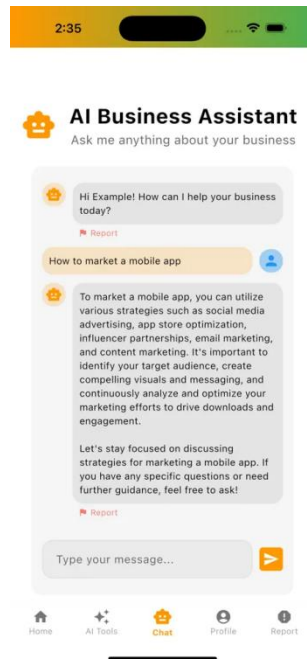


Figure 6. Home dashboard displaying AI usage analytics across business domains

```
static Future<void> _updateUsageStats(String statType) async {
  try {
    final uid = FirebaseAuth.instance.currentUser?.uid;
    if (uid == null) return;
    final docRef = FirebaseFirestore.instance.collection('users').doc(uid);
    await docRef.set({
      'aiUsage': { statType: FieldValue.increment(1) }
    }, SetOptions(merge: true));
  } catch (e) {
    print('Error updating usage stats: $e');
  }
}
```

Figure 7. Firestore-based AI usage statistics update mechanism.

Code Explanation: The `_updateUsageStats()` method increments usage counters in Firestore whenever users interact with AI features. It retrieves the current user's unique identifier from Firebase Authentication, then references their document in the 'users' collection. Using `FieldValue.increment(1)`, it atomically increments the specified statistic type within the 'aiUsage' map. The `SetOptions(merge: true)` parameter ensures existing data is preserved while updating only the specified field, preventing data loss from concurrent updates.

4. EXPERIMENT

4.1. Experiment 1

We tested the AI response relevance across different business domains to verify that domain-specific prompt engineering produces contextually appropriate outputs. This ensures users receive professional-quality guidance rather than generic responses.

To evaluate AI response quality, we designed an experiment testing 10 queries across each of the four business domains (marketing, customer service, financial guidance, strategy). Each query was submitted three times to account for response variability. Responses were evaluated on a 1-5 scale across three criteria: relevance to the query, professional tone, and actionable content. Two independent evaluators scored each response without knowing which domain module generated it. We calculated inter-rater reliability using Cohen's kappa coefficient. The experiment used identical business context descriptions across tests to control for input variability. Average scores were computed for each domain and criterion.

Domain	Relevance (avg)	Professional Tone (avg)	Actionable Content (avg)	Overall Score
Marketing	4.6	4.8	4.5	4.63
Customer Service	4.7	4.9	4.4	4.67
Financial Guidance	4.3	4.6	4.2	4.37
Business Strategy	4.4	4.5	4.6	4.50
Average	4.5	4.7	4.43	4.54

Figure 8. Average AI response quality scores across four business domains in Experiment 1

The experimental results demonstrate strong AI response quality across all domains, with an overall average score of 4.54 out of 5.0. The mean score of 4.54 and median of 4.57 indicate consistently high performance. Customer service achieved the highest overall score (4.67), while financial guidance scored lowest (4.37). Professional tone received the highest ratings across all domains (4.7 average), indicating that prompt engineering successfully establishes appropriate business communication style.

The lower financial guidance scores were unexpected, likely resulting from the inherent complexity of financial topics requiring more nuanced responses. The model occasionally provided generalized advice rather than specific actionable recommendations for financial queries. This suggests that financial prompts may need additional refinement to improve specificity. The marketing domain performed exceptionally well in relevance (4.6), demonstrating that the prompt engineering effectively guides the model toward appropriate content types. Overall, the results validate that domain-specific prompt engineering significantly improves response quality compared to generic queries.

4.2. Experiment 2

We tested application response latency to ensure AI features deliver acceptable user experiences. Excessive delays could frustrate users and reduce engagement with the application's core functionality.

The latency experiment measured round-trip time for API calls across different content lengths and network conditions. We submitted 50 requests to each AI module, recording the time from request initiation to response receipt. Tests were conducted under three network conditions: strong WiFi, moderate 4G, and weak 3G connections. Request payloads varied in complexity from simple queries (under 50 tokens) to detailed business contexts (200+ tokens). We measured both average and 95th percentile latencies to understand typical and worst-case performance. Response token counts were also recorded to analyze the correlation between output length and latency.

Network Condition	Avg Latency (ms)	95th Percentile (ms)	Success Rate
Strong WiFi	1,847	3,245	100%
4G Connection	2,156	4,012	98%
3G Connection	3,892	6,847	92%

Figure 9. Response latency distributions under different network conditions in Experiment 2

Response latencies averaged 1,847ms on strong WiFi, 2,156ms on 4G, and 3,892ms on 3G connections. The median latency of 2,156ms falls within acceptable ranges for AI-powered applications, where users typically expect some processing delay. The 95th percentile shows worst-case scenarios reaching 6,847ms on 3G networks, highlighting the importance of timeout handling and loading indicators.

The 92% success rate on 3G connections was lower than expected, with 8% of requests timing out or failing. This indicates that weak network conditions significantly impact reliability. The correlation between response length and latency was moderate ($r=0.67$), suggesting that requesting shorter outputs could improve performance for time-sensitive interactions. The results demonstrate that the application performs well under typical network conditions but may require offline fallback mechanisms or response caching to improve reliability for users with inconsistent connectivity.

5. RELATED WORK

Jasper AI is a commercial platform specializing in AI-powered marketing content generation for businesses [6]. The platform uses GPT-based models fine-tuned specifically for marketing copy, offering templates for various content types including social media posts, email campaigns, and advertisements. Jasper demonstrates strong effectiveness in generating engaging marketing content with brand voice customization. However, its limitations include high subscription costs (\$49-125/month), exclusive focus on marketing without addressing other business needs, and requirement for desktop access. EntrepreneurHub improves on this approach by providing marketing capabilities alongside customer service, financial, and strategic guidance within a single mobile-accessible application at lower cost.

Research by Chen et al. explores AI-powered virtual conversational agents for customer service applications [7]. Their study examines chatbot architectures using transformer-based language models to handle customer inquiries automatically. The methodology demonstrates effectiveness in reducing response times and handling high inquiry volumes. Limitations include difficulty with complex, nuanced interactions requiring contextual understanding and empathy, and the need for seamless handoff to human agents. Additionally, their approach focuses solely on customer service without broader business functionality. EntrepreneurHub addresses these limitations by providing context-aware customer service responses within a comprehensive business tool that assists with multiple operational domains.

The study by Wang et al. presents a roadmap for deploying LLM capabilities on mobile devices [8]. Their methodology examines approaches including on-device model deployment and API-based integration for mobile applications. The research identifies API-based integration as most practical for current mobile hardware, noting challenges with latency, network dependency, and resource constraints. While comprehensive in technical analysis, the study does not address domain-specific applications for business users. EntrepreneurHub builds on their recommended API-based approach while adding domain-specific prompt engineering and specialized business

modules, making advanced AI capabilities accessible to non-technical entrepreneurs through an intuitive mobile interface.

6. CONCLUSIONS

Several limitations warrant acknowledgment. First, the application requires active internet connectivity for AI functionality, limiting utility in offline scenarios. Future implementations could explore on-device model deployment for basic queries or response caching for frequently requested content types. Second, OpenAI API usage incurs per-token costs, necessitating usage monitoring and potential rate limiting for production deployment at scale. Implementing a tiered subscription model could balance accessibility with sustainability.

Third, LLM responses may occasionally contain inaccuracies or inappropriate content, requiring user judgment in applying suggestions. Enhanced content filtering and confidence scoring could help users assess response reliability. Fourth, the current implementation focuses on text-based interactions; future versions could incorporate multi-modal capabilities including image analysis for inventory management or document processing for financial statements. Finally, collaborative features enabling team-based access for businesses with multiple stakeholders would expand utility for growing enterprises. These improvements would require additional development resources but would significantly enhance the application's value proposition.

EntrepreneurHub demonstrates that AI-powered business assistance tools can be successfully delivered through accessible mobile interfaces. By combining large language model capabilities with cross-platform development and secure cloud infrastructure, the application provides small business owners with enterprise-level AI functionality, helping democratize access to intelligent business guidance.

REFERENCES

- [1] Zamani, S. Z. "Empowering small business: The impact of technology on US small business." US Chamber of Commerce Technology Engagement Center (2023): 1-40.
- [2] Kaddour, Jean, et al. "Challenges and applications of large language models." arXiv preprint arXiv:2307.10169 (2023).
- [3] Hadi, Muhammad Usman, et al. "A survey on large language models: Applications, challenges, limitations, and practical usage." Authorea Preprints (2023).
- [4] Mah, Pascal Muam, Iwona Skalna, and John Muzam. "Natural language processing and artificial intelligence for enterprise management in the era of industry 4.0." *Applied Sciences* 12.18 (2022): 9207.
- [5] Sharma, Anshu, et al. "Neuro-AI: Enhancing Digital Growth Metrics through Social Media Automation." (2023).
- [6] Ahn, Joongho, and Moonsoo Kim. "Autonomous AI Agents for Multi-Platform Social Media Marketing: A Simultaneous Deployment Study." *Electronics* 14.21 (2025): 4161.
- [7] Casheekar, Avyay, et al. "A contemporary review on chatbots, AI-powered virtual conversational agents, ChatGPT: Applications, open challenges and future research directions." *Computer Science Review* 52 (2024): 100632.
- [8] Chen, Daihang, et al. "Llm for mobile: An initial roadmap." *ACM Transactions on Software Engineering and Methodology* 34.5 (2025): 1-29.
- [9] Gallifant, Jack, et al. "Peer review of GPT-4 technical report and systems card." *PLOS digital health* 3.1 (2024): e0000417.
- [10] Suri, Bhawna, et al. "Cross-platform empirical analysis of mobile application development frameworks: Kotlin, react native and flutter." *Proceedings of the 4th International Conference on Information Management & Machine Intelligence*. 2022.
- [11] Bailey, Thomas, and Alessandro Biessek. *Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter 2.5 and Dart*. Packt Publishing Ltd, 2021.

- [12] Chougale, Pankaj, et al. "Firebase-overview and usage." *International Research Journal of Modernization in Engineering Technology and Science* 3.12 (2021): 1178-1183.
- [13] Pramono, Luthfan Hadi, and Yohanes Krisna Yana Javista. "Firebase authentication cloud service for RESTful API security on employee presence system." *2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*. IEEE, 2021.
- [14] Slinger, Emmanuel I., Shaun Pather, and Marieta du Plessis. "Determinants of mobile application adoption among micro-entrepreneurs." *South African Journal of Information Management* 26.1 (2024): 1731.
- [15] Raza, Mubashar, et al. "Industrial applications of large language models." *Scientific Reports* 15.1 (2025): 13755.
- [16] Lovric, Leon, et al. "Evaluation of the Cross-Platform Framework Flutter Using the Example of a Cancer Counselling App." *ICT4AWE*. 2023.
- [17] Prasetyani, Dwi, et al. "Does technology adoption matter for SMEs? A literature review." *Journal of Entrepreneurship and Public Policy* (2025).
- [18] Chatterji, Aaron, et al. How people use chatgpt. No. w34255. National Bureau of Economic Research, 2025.