Vol. 6 No. 2, 2025, pp. 120-130 **Jurnal Pepadun**

Development of Virtual Assistant using OpenAI Assistant for Immigration Services

[*1]Shalahuddin Abdul Aziz, [2]Rico Andrian, [3]Muhaqiqin, and [4]Admi Syarif

[1,2,3,4]Department of Computer Science, University of Lampung,
Jalan Prof. Dr. Sumantri Brojonegoro Nomor 1, Bandar Lampung, Indonesia
e-mail: [*1]**shalahuddin.abdul21@students.unila.ac.id**, [2]rico.andrian@fmipa.unila.ac.id, [3]muhaqiqin@fmipa.unila.ac.id, [4]admi.syarif@fmipa.unila.ac.id

Abstract *- The development of technology in the fields of Artificial Intelligence (AI) and Machine Learning (ML) has resulted in technologies such as Large Language Models (LLM). LLM is a technology that enables humans to communicate with computers in a natural manner. This technology has the potential to replace customer service in many sectors by utilizing Virtual Assistants based on LLM. This research aims to develop a virtual assistant that can replace customer service and address the Frequently Asked Questions of the Directorate General of Immigration. The study was conducted at the Department of Computer Science, University of Lampung, from September 2024 to January 2025. The technologies used include OpenAI Assistant along with LLM GPT-4o, OpenAI Whisper, Next.js 14, MongoDB, and Typescript. The knowledge base for the virtual assistant utilizes Constitution No. 6 of 2011 as the foundational knowledge to ensure that the answers provided are in accordance with applicable regulations and laws in Indonesia. The methodology applied in this research is Extreme Programming (XP), which involves phases of planning, design, coding, testing, and listening. The developed virtual assistant demonstrated significant improvements in providing quick and accurate immigration information compared to traditional FAQ systems, enhancing user satisfaction and accessibility.*

Keywords**:** *OpenAI Assistant; Directorate General of Immigration; Extreme Programming; Large Language Models; Retrieval Augmented Generation.*

1. INTRODUCTION

In the current digital era, the interaction between humans and technology has become an integral part of daily life. Artificial Intelligence (AI) and Large Language Models (LLMs) have led to significant breakthroughs, particularly in the form of Virtual Assistants (VAs). These systems are designed to simulate human conversation, providing information and assistance to users in a natural, intuitive manner [1]. VAs have immense potential to revolutionize customer service across various sectors by automating responses and improving efficiency [2][3].

The Directorate General of Immigration (Ditjenim) of Indonesia is responsible for managing immigration policies, including travel documents like visas and residence permits. Currently, the public primarily interacts with Ditjenim through its official website or by visiting immigration offices. However, this system presents several challenges. The website's Frequently Asked Questions (FAQ) section is often inadequate, failing to provide comprehensive or easily navigable answers. This forces users to rely on human agents, who are only available during working hours and can be overwhelmed by the high volume of repetitive inquiries. These issues are common across many e-government services, often leading to long waiting times and a negative user experience[4][5][6]. Long waiting times and unresponsive systems are major pain points in customer service [7].

This research proposes the development of a web-based Virtual Assistant to serve as a primary information channel for Indonesian immigration services. By leveraging the OpenAI Assistant API, this VA can provide 24/7 support, answering user queries accurately and consistently [8]. The system is grounded in a specific knowledge base, primarily Law No. 6 of 2011 on Immigration and official Ditjenim publications, ensuring the information provided is reliable and up-to-date [9]. The goal is to create a more efficient, accessible, and user-friendly channel for immigration-related queries.

 | doi: 10.23960/pepadun.v6i3.272

The application of VAs and chatbots in customer service and specialized domains has been explored extensively. Evaluations ChatGPT-4's accuracy in dental diagnostics, highlights its potential but also the need for expert supervision due to inconsistencies [10]. In the context of customer service chatbots, the implementation of Mi- crosoft's LUIS and QnA Maker, using a knowledge base for accurate responses and LLMs for natural interaction [11]. The OpenAI Assistant API, central to this work, allows for the creation of sophisticated VAs by integrating LLMs (like GPT-4o) with tools such as file search Retrieval Augmented Generation (RAG) and code interpreters [8]. The use of OpenAI Whisper for speech-to-text and text-to-speech further enhances accessibility [12][13].

The Extreme Programming (XP) methodology was chosen for this project due to its flexibility, iterative nature, and emphasis on user feedback [14]. This approach is well-suited for a project where requirements may evolve, allowing for rapid development cycles and continuous refinement of the system based on real user interactions. The goal of this research is to develop a VA that enhances the convenience, responsiveness, and trustworthiness of immigration information services, thereby improving overall public satisfaction.

2. RESEARCH METHODOLOGY

This study employed the Extreme Programming (XP) methodology, an agile framework that focuses on delivering software through iterative development cycles [15]. The XP process used in this research consisted of five main phases, which were repeated across several iterations: Planning, Design, Coding, Testing, and Listening. This cyclical approach allowed for continuous integration of feedback and incremental feature development. The overall research workflow is depicted in Figure 1.

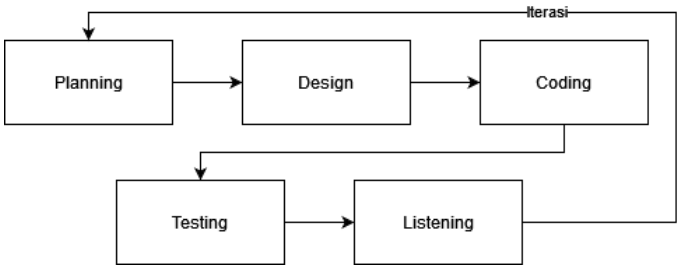

Figure 1. Extreme Programming (XP).

2.1 Planning

The planning phase involved identifying user needs and defining the project scope. User stories were created based on initial discussions with stakeholders to outline functional requirements. These stories guided the development priorities for each iteration.

2.2 Design

The design phase designs system architecture and user interface. Key artifacts included Use Case Diagrams, Entity-Relationship Diagrams (ERD) for the database schema, and low-fidelity prototypes for the UI/UX. The design focused on simplicity and functionality.

2.3 Coding

In this phase, the designs were translated into a functional application. The development stack included Next.js 14 for its modern, server-side rendering capabilities [16][17]. Typescript is used instead of just Javascript to ensure code robustness and scalability [18]. MongoDB was used as the NoSQL database due to its flexible, document-based structure suitable for chat applications [19], and user authentication was managed with NextAuth.js, a secure framework for handling credentials and sessions [20].

 | doi: 10.23960/pepadun.v6i3.272

2.4 Testing

Each feature was subjected to rigorous testing to ensure it met the acceptance criteria defined in the planning phase. Black-box testing was conducted to verify functionality from an end-user perspective without knowledge of the internal code. User surveys were distributed via Google Forms to gather quantitative feedback on usability and satisfaction.

2.5 Listening (Feedback)

This phase focused on gathering and analyzing qualitative feedback from end-users. This feedback was crucial for identifying issues, understanding user pain points, and planning improvements for subsequent iterations. The project was executed over five development iterations, each building upon the last to deliver a more refined and feature-rich application.

3. RESULTS AND DISCUSSION

This section outlines the results of the development process, including the system architecture, the key features implemented in each iteration, and the final evaluation results.

3.1 System Architecture

The virtual assistant operates on a client-server architecture deployed on Vercel. As shown in Figure 2, the user interacts with the Next.js frontend. Authentication is handled by NextAuth.js, which verifies user credentials against the MongoDB database. Once authenticated, user prompts are sent to the backend, which securely calls the OpenAI Assistant API. The assistant, equipped with the immigration knowledge base, processes the prompt and generates a response. The chat history is stored in MongoDB, allowing for persistent conversations.

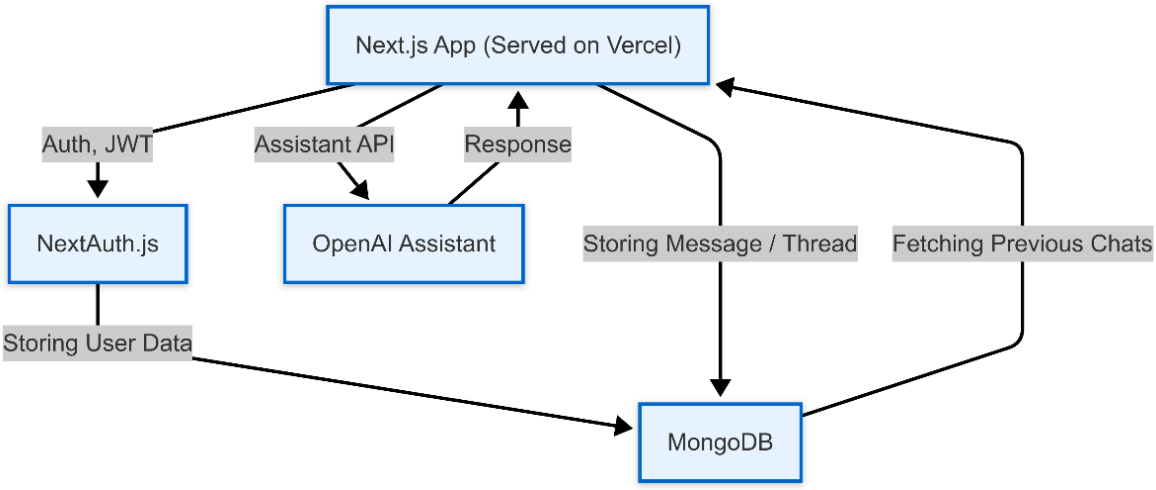

Figure 2. System architecture diagram.

3.2 Development Iterations and Features

The application was developed incrementally across five iterations:

3.2.1 Iteration 1

 | doi: 10.23960/pepadun.v6i3.272

The first iteration focused on establishing the foundational components. This included setting up user authentication (registration and login) with NextAuth.js and MongoDB, and implementing the basic chat interface [19] [20]. An initial integration with the OpenAI Assistant API was completed to enable fundamental conversational capabilities, leveraging the underlying GPT model as a powerful coding and response generation tool [8].

```
import mongoose from 'mongoose';
const UserSchema = new mongoose.Schema({
  name: { type: String, required: [true, 'Please provide a name'], maxlength: [60, 'Name cannot be
more than 60 characters'],
  },
  email: { type: String, required: [true, 'Please provide an email'], unique: true, lowercase: true,
    validate: { validator: function (v: string) { return /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(v); },
      message: 'Please enter a valid email', }, },
  password: { type: String, required: [true, 'Please provide a password'], minlength: [8, 'Password
must be at least 8 characters long'], },
  createdAt: { type: Date, default: Date.now, }, });
export default mongoose.models.User || mongoose.model('User', UserSchema);
```

Figure 3. User schema.

The schema in Figure 3 enforces strict validation rules to ensure data integrity and security, which is critical for the authentication system. It defines the essential user attributes: a required name with a maximum length, a password with a minimum length of eight characters for basic security, and an email field that is mandatory, unique, and automatically converted to lowercase to ensure consistency during lookups. This robust schema serves as the data backbone for the authentication features handled by NextAuth.js, ensuring that only valid and well-formed user data is persisted in the database.

3.2.2 Iteration 2

```
> Anda adalah chatbot imigrasi Indonesia berbasis yang bertugas memberikan informasi akurat tentang layanan
imigrasi. Anda dirancang untuk membantu warga negara Indonesia dan turis asing dengan pertanyaan umum:
> * Cara membuat paspor untuk warga negara Indonesia.
> * Proses pengajuan visa ke negara asing untuk warga Indonesia.
> * Prosedur mendapatkan Visa on Arrival (VoA) untuk turis asing yang ingin berkunjung ke Indonesia.
> **Aturan Penting**:
> * Gunakan informasi yang terdapat dalam data yang telah disediakan kepada Anda sebagai sumber utama. Jangan
menyimpulkan atau menambahkan informasi yang tidak tercantum. Berikan penjelasan yang ramah, rinci, dan jelas.
> * Jangan menjawab pertanyaan di luar topik imigrasi. Jika pengguna bertanya tentang hal lain, dengan sopan
arahkan mereka ke lembaga atau sumber informasi yang relevan. Tanggapi dengan sopan dan beri tahu bahwa Anda
hanya dapat menjawab pertanyaan terkait layanan imigrasi.
> * Jelaskan bahwa kebijakan imigrasi dapat berubah sewaktu-waktu, sehingga pengguna disarankan untuk selalu
memeriksa informasi terkini dari sumber resmi seperti [www.imigrasi.go.id](https://www.imigrasi.go.id).
> **Tugas Anda adalah memberikan informasi imigrasi sesuai dengan dokumen yang disediakan. Jangan
memberikan informasi di luar cakupan dokumen atau imigrasi.
> **Contoh Pertanyaan yang Diharapkan**:
> * "Bagaimana cara membuat paspor baru untuk WNI?"
> * "Dokumen apa saja yang dibutuhkan untuk mengajukan visa ke Jepang?"
> * "Apa saja syarat Visa on Arrival untuk masuk ke Indonesia?"
```

Figure 4. System prompt.

This crucial iteration involved grounding the VA's responses in factual data. A detailed system prompt was engineered to instruct the AI on its role, limitations, and response format. The primary knowledge base, including Law No. 6 of 2011 and official FAQs, was provided to the OpenAI Assistant using its file search

 | doi: 10.23960/pepadun.v6i3.272

(RAG) capabilities. This technique is crucial for grounding the model's responses in factual data to improve accuracy and relevance in the context of specialized virtual assistants and for customer service chatbots [9][11].

Figure 4 shows detailed system prompt was engineered to define the virtual assistant's persona, operational boundaries, and response protocol. The prompt explicitly instructs the AI to act as an official Indonesian immigration chatbot, tasked with providing accurate information to both Indonesian citizens and foreign tourists on topics such as passport creation, visa applications, and Visa on Arrival (VoA) procedures. A critical aspect of the prompt is the strict instruction to ground all responses exclusively in the provided knowledge base (Law No. 6 of 2011 and official FAQs), thereby minimizing the risk of factual inaccuracies or "hallucinations." Furthermore, it establishes clear guardrails by directing the assistant to politely decline any queries outside the scope of immigration. Finally, the prompt defines a professional yet friendly tone, includes a disclaimer about the dynamic nature of immigration policies, and mandates a standardized greeting, ensuring a consistent and responsible user experience.

3.2.3 Iteration 3

To enhance user experience, a chat history feature was implemented. Schemas for thread and message were created in MongoDB, taking advantage of its flexible document model which, is well-suited for the JSON-like structure of conversational data [19]. This allows users to review past interactions and resume previous conversations.

```
import mongoose, { Schema } from 'mongoose';
export enum MessageRole {
  User = 'user', Assistant = 'assistant', }
const MessageSchema: Schema = new Schema(
  { id: { type: String, required: true, unique: true },
    role: { type: String, enum: MessageRole, required: true },
    content: { type: String, required: true },
    thread: { type: mongoose.Schema.Types.ObjectId, ref: 'Thread', },
  },
  { timestamps: true }
);
export default mongoose.models.Message || mongoose.model('Message', MessageSchema);
```

Figure 5. Message schema.

```
import mongoose, { Schema } from 'mongoose';
const ThreadSchema: Schema = new Schema(
  { threadId: { type: String, required: true },
    userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true, },
    messages: [ { type: mongoose.Schema.Types.ObjectId, ref: 'Message', }, ], },
  { timestamps: true, }
);
ThreadSchema.index({ userId: 1 });
export default mongoose.models.Thread || mongoose.model('Thread', ThreadSchema);
```

Figure 6. Thread schema.

Figure 5 and Figure 6 implements the chat history feature, a relational data model was designed using two distinct MongoDB schemas: Thread and Message. The ThreadSchema serves as the primary container for a single conversation, establishing ownership by linking to a specific userId and holding an array of references to all associated messages. The MessageSchema represents each individual utterance within a thread, storing the text content and a crucial role property (user or assistant) to distinguish the speaker. This one-to-many

 | doi: 10.23960/pepadun.v6i3.272

structure, where each message references its parent thread, creates a robust and efficient system for storing and retrieving entire conversational histories, ensuring a persistent and seamless user experience.

3.2.4 Iteration 4

This iteration introduced speech capabilities. Speech-to-text was implemented using the browser's Web Speech API, allowing users to speak their questions. Text-to-speech was added using the OpenAI Whisper API to voice the assistant's responses, making the interaction more natural and accessible [13]. This aligns with the vision for creating speech-enabled environments which corelates with the persuasive influence of the voice modality in virtual assistants [12][21].

Figure 7 shows the logic that runs the WebSpeech component. The component's logic is built around the browser's native Web Speech API. Upon mounting, a useEffect hook initializes a SpeechRecognition instance, first checking for browser compatibility to ensure graceful degradation. The instance is configured for the Indonesian language (id-ID) and set to provide continuous, interim results, which is essential for the real-time transcript display. The component's behavior is event-driven: the on-result event handler captures speech and updates the transcript state, while the on-error and on-end handlers manage errors and the recording lifecycle.

```
const WebSpeech: React.FC<WebSpeechProps> = ({ onTranscript }) => {
  const [isListening, setIsListening] = useState(false);  const [transcript, setTranscript] = useState('');
  const [error, setError] = useState<string | null>(null);
  useEffect(() => {let recognition: SpeechRecognition | null = null;
    if ('webkitSpeechRecognition' in window) {
      recognition = new ( window as any ).webkitSpeechRecognition() as SpeechRecognition;
      recognition.lang = 'id-ID'; recognition.continuous = true; recognition.interimResults = true;
      recognition.onresult = (event: SpeechRecognitionEvent) => {
        const current = event.resultIndex; const transcriptText = event.results[current][0].transcript;
        setTranscript(transcriptText); };
      recognition.onerror = (event: SpeechRecognitionErrorEvent) => {
        setError(`Speech recognition error: ${event.error}`); };
      recognition.onend = () => { setIsListening(false);};
      (window as any).recognition = recognition;
    } else { setError('Your browser does not support speech recognition.'); }
    return () => { if (recognition) recognition.stop(); };  }, []);
  const toggleListening = (isSend: boolean) => {
    const recognition = (window as any).recognition as | SpeechRecognition | undefined;
    if (recognition) {
      if (isListening) { recognition.stop();
      } else { setTranscript(''); recognition.start();
      } setIsListening(!isListening);
    } if (isSend) handleSubmit();
  };  const handleSubmit = () => { onTranscript(transcript); setTranscript(''); };};
```

Figure 7. WebSpeech logic.

3.2.5 Iteration 5

Based on feedback from the Listening phase—a core tenet of the Extreme Programming methodology as—this final iteration addressed usability issues, particularly on mobile devices [14]. The UI was made fully responsive, with a collapsible sidebar and improved layout for smaller screens to ensure a consistent experience across all devices.

 | doi: 10.23960/pepadun.v6i3.272

```
const Header = ({ threadId, timestamp }: Props) => {
  const { currentThreadId } = useThread();
  return (
    <header className="flex justify-between bg-gray-700 rounded-md m-4 p-4">
      {!currentThreadId ? ( <p>{threadId ? threadId : 'Start by asking a question!'}</p>  ) : (
        <p>Thread Created On: {formatTimestamp(timestamp)}</p>
      )} <SidebarTrigger />
    </header>
  );
};
export default Header;
```

Figure 8. Header component code.

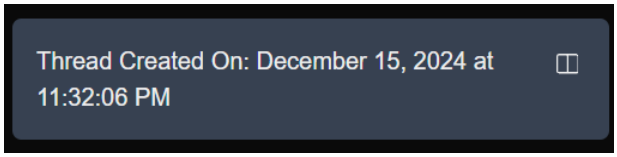

Figure 9. Header component.

The Header component in Figure 9 serves as the dynamic title bar for the chat interface, providing users with contextual information about the current conversation. It conditionally displays either a default prompt for new chats or the creation timestamp for an active thread. Crucially, it also integrates the SidebarTrigger component, which provides a control for users to toggle the visibility of the chat history sidebar—a key feature for ensuring a responsive layout on smaller screens.

3.3 Evaluation Results

To evaluate the effectiveness of the virtual assistant, a comparative analysis was conducted against the traditional method of finding information on the Ditjenim FAQ page. This approach aligns with studies evaluating other e-government chatbots, which also measure improvements in efficiency and user satisfaction against legacy systems [6]. Eleven end-users were asked to find answers to five common immigration questions using both methods.

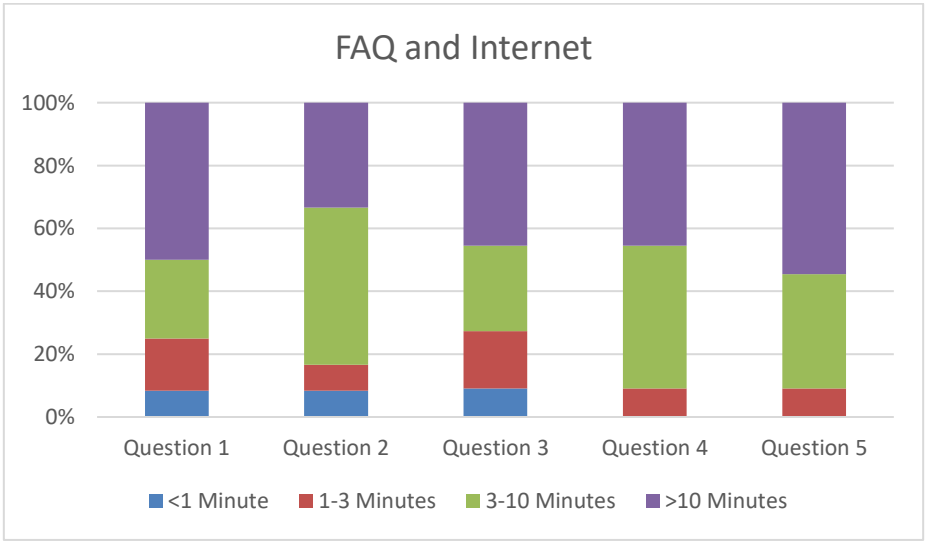

Figure 10. FAQ and internet response time.

 | doi: 10.23960/pepadun.v6i3.272

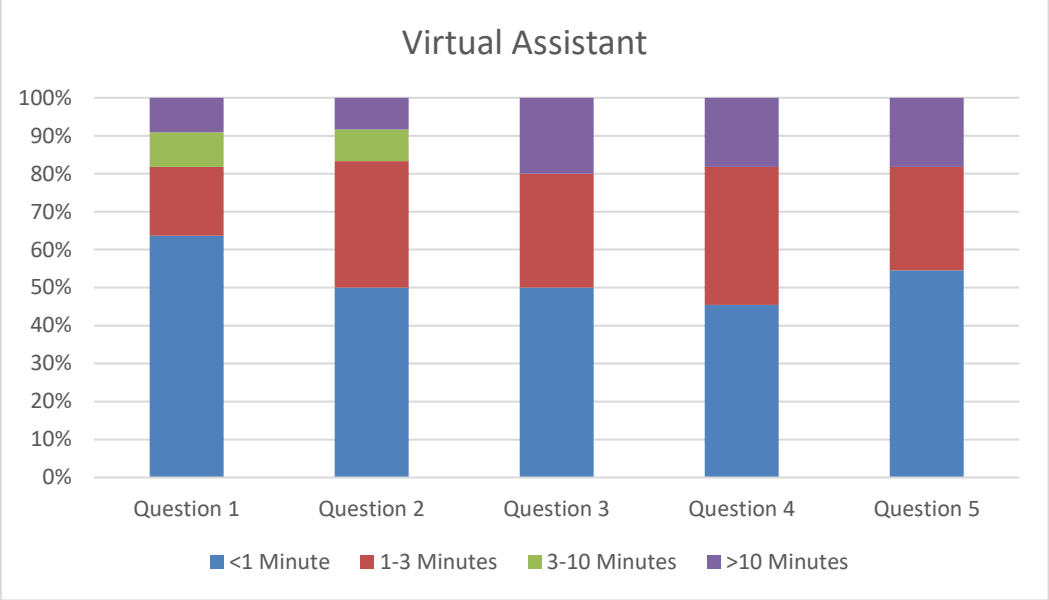

Figure 11. VA response time.

The results, visualized in Figure 10 and Figure 11, show a dramatic improvement in efficiency, a finding that is consistent with the demonstrated performance of other NLP-powered government assistants [6]. With the VA, most users found answers in under three minutes. In contrast, searching the FAQ page took the majority of users over ten minutes, with many expressing frustrations.

Table 1. Qualitative comparison of information retrieval methods.

Aspect	**Before VA (Using FAQ and Internet)**	**After VA (Using VA)**
Speed & Context	Slow; requires reading long texts and manually synthesizing information.	Fast and conversational; provides direct answers, saving users from reading extensive documents.
Convenience	Requires active searching via a search engine or navigating a complex site.	Easily accessible as a dedicated tool on the immigration website or app.
Availability	Information is online 24/7, but finding the correct, up-to-date document is difficult.	The VA is available 24/7 and provides curated, reliable information instantly.
Quality & Consistency	Information on the internet can be outdated, inaccurate, or unofficial.	Responses are sourced from a controlled, official knowledge base, ensuring quality and consistency.

The evaluation demonstrates that the developed Virtual Assistant successfully addresses the shortcomings of the existing system. The qualitative comparison in The results, visualized in Figure 10 and Figure 11, show a dramatic improvement in efficiency, a finding that is consistent with the demonstrated performance of other NLP-powered government assistants [6]. With the VA, most users found answers in under three minutes. In contrast, searching the FAQ page took the majority of users over ten minutes, with many expressing frustrations.

Table 1 highlights these improvements across several key aspects. In terms of Speed and Convenience, the VA provides direct, conversational answers, saving users from the effort of navigating complex websites and manually synthesizing information. Regarding Quality and Consistency, the VA ensures that responses are reliable and

 | doi: 10.23960/pepadun.v6i3.272

consistent by drawing from an official, controlled knowledge base, a significant advantage over potentially outdated or inaccurate information found on the wider internet. This provides a faster, more convenient, and more reliable channel for the public to access immigration information.

4. CONCLUSIONS

This study successfully developed a web-based Virtual Assistant for Indonesian immigration services using the Extreme Programming methodology and the OpenAI Assistant API. The research demonstrates that such a system can significantly enhance public service quality by providing an efficient, accessible, and reliable information channel. The iterative XP approach proved effective, allowing for the incremental addition of complex features like a grounded knowledge base and multimodal interaction while continuously incorporating user feedback. The final evaluation confirmed that the VA drastically reduces the time and effort required for users to obtain accurate immigration information compared to traditional methods. The high user satisfaction ratings underscore its potential as a valuable tool for Ditjenim.

For future work, it is recommended to explore the use of open-source, on-premise LLMs to ensure full data privacy and variety of options with LLMs fine-tuned specifically for RAG pipeline. Additionally, implementing a more advanced, manually configured RAG pipeline using frameworks like LangChain or Haystack could offer greater control and flexibility over the knowledge retrieval process.

LITERATURE

[1] D. Bernard and A. Arnold, “Cognitive interaction with virtual assistants: From philosophical foundations to illustrative examples in aeronautics,” *Comput Ind*, vol. 107, pp. 33–49, May 2019, doi: 10.1016/j.compind.2019.01.010.

[2] E. al. Audi Albtoush, “ChatGPT: Revolutionizing User Interactions with Advanced Natural Language Processing,” *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 11, no. 9, pp. 3354–3360, Nov. 2023, doi: 10.17762/ijritcc.v11i9.9541.

[3] J. Trivedi, “Examining the Customer Experience of Using Banking Chatbots and Its Impact on Brand Love: The Moderating Role of Perceived Risk,” *Journal of Internet Commerce*, vol. 18, no. 1, pp. 91–111, Jan. 2019, doi: 10.1080/15332861.2019.1567188.

[4] Y. Zhao, T. Zhang, Y. Liu, Y. Zhu, and Y. Gao, “Research on the Influence Mechanism of Artificial Intelligence(AI) Customer Service on User Satisfaction with Online Shopping,” in *2021 2nd International Conference on Computer Science and Management Technology (ICCSMT)*, IEEE, Nov. 2021, pp. 253–260. doi: 10.1109/ICCSMT54525.2021.00056.

[5] P. D. A. Mahendra, K. A. S. Wijaya, and I. K. Winaya, “Optimalisasi Layanan M-Paspor Dari Sudut Pandang Responsiveness dan Reliability di Kantor Imigrasi Denpasar,” *Sawala : Jurnal Administrasi Negara*, vol. 12, no. 1, pp. 229–239, Jun. 2024, doi: 10.30656/SAWALA.V12I1.8133.

[6] M. M. Siahaan, R. A. Sunarjo, R. Sebastian, and S. M. Wahid, “The Role of Natural Language Processing in Enhancing Chatbot Effectiveness for E-Government Services,” *Journal of Computer Science and Technology Application*, vol. 2, no. 1, pp. 65–74, Mar. 2025, doi: 10.33050/754QC238.

[7] Y. Xu, C.-H. Shieh, P. van Esch, and I.-L. Ling, “AI Customer Service: Task Complexity, Problem-Solving Ability, and Usage Intention,” *Australasian Marketing Journal*, vol. 28, no. 4, pp. 189–199, Nov. 2020, doi: 10.1016/j.ausmj.2020.03.005.

[8] L. Moussiades and G. Zografos, “OpenAi’s GPT4 as coding assistant,” Sep. 2023, Accessed: Oct. 30, 2024. [Online]. Available: https://arxiv.org/abs/2309.12732v1

[9] C. Wibhowo and R. Sanjaya, “Virtual Assistant to Suicide Prevention in Individuals with Borderline Personality Disorder,” in *2021 International Conference on Computer & Information Sciences (ICCOINS)*, IEEE, Jul. 2021, pp. 234–237. doi: 10.1109/ICCOINS49721.2021.9497160.

[10] A. Suárez, J. Jiménez, M. Llorente de Pedro, C. Andreu-Vázquez, V. Díaz-Flores García, M. Gómez Sánchez, and Y. Freire, “Beyond the Scalpel: Assessing ChatGPT’s potential as an auxiliary intelligent virtual assistant in oral surgery,” *Comput Struct Biotechnol J*, vol. 24, pp. 46–52, Dec. 2024, doi: 10.1016/j.csbj.2023.11.058.

[11] C.-C. Chang, W.-S. Cheng, and S. Hsiao, “Customer Service Chatbot Enhanced with Conversational Language Understanding and Knowledge Base,” in *2022 IEEE 4th Eurasia Conference on IOT, Communication and Engineering (ECICE)*, IEEE, Oct. 2022, pp. 231–234. doi: 10.1109/ECICE55674.2022.10042940.

[12] G. Iannizzotto, L. Lo Bello, A. Nucita, and G. M. Grasso, “A vision and speech enabled, customizable, virtual assistant for smart environments,” *Proceedings - 2018 11th International Conference on Human System Interaction, HSI 2018*, pp. 50–56, Aug. 2018, doi: 10.1109/HSI.2018.8431232.

[13] C. Ischen, T. B. Araujo, H. A. M. Voorveld, G. Van Noort, and E. G. Smit, “Is voice really persuasive? The influence of modality in virtual assistant interactions and two alternative explanations,” *Internet Research*, vol. 32, no. 7, pp. 402–425, Dec. 2022, doi: 10.1108/INTR-03-2022-0160.

[14] R. Juric, “Extreme programming and its development practices,” in *ITI 2000. Proceedings of the 22nd International Conference on Information Technology Interfaces (Cat. No.00EX411)*, 2000, pp. 97–104.

[15] S. Krishna and K. Tadikonda, “Corresponding author: Satya Krishna Kapil Tadikonda Bridging disciplines: Cross-functional collaboration frameworks in modern AI Development,” *World Journal of Advanced Engineering Technology and Sciences*, vol. 2025, no. 01, pp. 203–210, 2025, doi: 10.30574/wjaets.2025.15.1.0211.

[16] M. Thakkar, “Building React Apps with Server-Side Rendering: Use React, Redux, and Next to Build Full Server-Side Rendering Applications,” *Building React Apps with Server-Side Rendering: Use React, Redux, and Next to Build Full Server-Side Rendering Applications*, pp. 1–192, Jan. 2020, doi: 10.1007/978-1-4842-5869-9.

[17] H. H. Ben kora and M. S. Manita, “Modern Front-End Web Architecture Using React.js and Next.js,” *University of Zawia Journal of Engineering Sciences and Technology*, vol. 2, no. 1, pp. 1–13, Aug. 2024, doi: 10.26629/uzjest.2024.01.

[18] J. Scarsbrook, M. Utting, and R. Ko, “TypeScript’s Evolution: An Analysis of Feature Adoption Over Time,” Oct. 2023. doi: 10.48550/arXiv.2303.09802.

[19] R. Byali, Ms. Jyothi, and M. C. Shekadar, “"Evaluation of NoSQL Database MongoDB with Respect to JSON Format Data Representation ",” *International Journal of Research Publication and Reviews*, pp. 867–871, Sep. 2022, doi: 10.55248/gengpi.2022.3.9.24.

[20] A. Ezugwu, E. Ukwandu, C. Ugwu, M. Ezema, C. Olebara, J. Ndunagu, L. Ofusori, and U. Ome, “Password-based authentication and the experiences of end users,” *Sci Afr*, vol. 21, p. e01743, Sep. 2023, doi: 10.1016/J.SCIAF.2023.E01743.

 | doi: 10.23960/pepadun.v6i3.272

[21] W. Wei, S. Li, S. Okada, and K. Komatani, "Multimodal User Satisfaction Recognition for Non-task Oriented Dialogue Systems," in *Proceedings of the 2021 International Conference on Multimodal Interaction*, New York, NY, USA: ACM, Oct. 2021, pp. 586–594. doi: 10.1145/3462244.3479928.