

Enhancing Service-Oriented Architectures with Generative AI: A Case Study in Local Web-Based Service Discovery

¹Festim Halili*, ¹Enisa Abazi, ¹Merita Kasa Halili, ¹Halim Halimi, ²Enes Bajrami

Author's Affiliation:

¹Faculty of Natural Science and Mathematics, University of Tetova, North Macedonia

²Faculty of Computer Science and Engineering, Ss. Cyril and Methodious University, North Macedonia

*Correspondent author: festim.halili@unite.edu.mk

ABSTRACT

This paper studies the integration of Generative AI with Service-Oriented Architectures to enhance local web-based service discovery. As more and more organizations have started the process of adopting SOA for updating their software environments, the integration of Generative AI presents some exciting opportunities in improving state-of-the-art service matching and personalization. The case study herein demonstrates the use of Flask in tandem with a generative AI model to match user queries against services. With the system in this paper responding more to service-oriented computing through keyword detection and natural language processing of user input, initial testing presents the model as effective in the precise retrieval of services with a view to presenting a user-friendly interface for dynamic service discovery. This approach identifies the feasibility of AI-driven enhancements to SOA and creates the bedrock for subsequent applications in leveraging Generative AI for service delivery frameworks.

KEYWORDS: *Generative AI, Service Discovery, SOA, NLP, Web Services*

1. Introduction

Service-Oriented Architecture (SOA) has garnered increasing attention as organizations seek to modernize their software systems and address challenges in diverse operational environments [1] [2]. Transitioning from legacy platforms to SOA-based architectures has become a prevalent strategy, facilitating the integration of cutting-edge technologies such as the Internet of Things (IoT), Cloud Computing, and microservices [3]. SOA's modular structure promotes flexible integration and service reuse, offering a unified framework that consolidates various applications and data sources within a cohesive "black box." This approach ensures that IT resources remain accessible, regardless of underlying technologies, programming languages, or platforms [4] [5]. As the demand for innovative solutions grows, the integration of Generative AI into SOA presents exciting opportunities. Generative models, especially diffusion models, are at the forefront of AI advancements, enabling the creation of high-quality synthetic data across multiple modalities, including images, text, and audio [6] [7]. Originally developed for denoising images, diffusion models have evolved to effectively capture complex data distributions, making them versatile tools for various applications. While early generative models, such as Hidden Markov Models (HMMs) and Gaussian Mixture Models (GMMs), faced limitations due to their reliance on hand-designed features, the introduction of deep learning has led to significant improvements through models like Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) [8]. However, GANs often encountered architectural instabilities, prompting the shift toward diffusion models, which draw inspiration from non-equilibrium thermodynamics [9]. This approach increases entropy and randomness over time, enhancing the model's ability to generate diverse outputs. Recent innovations in diffusion models, particularly from OpenAI, have made them more practical for everyday applications, paving the way for their integration

within SOA frameworks [7].

2. Literature Review

Author in [10] highlights that recent advancements in Generative Artificial Intelligence (GenAI) tools have significantly impacted software development, providing valuable assistance across various managerial and technical project activities. Prominent examples of these tools include OpenAI's ChatGPT, GitHub Copilot, and Amazon CodeWhisperer. Despite the growing body of literature evaluating the application of GenAI in software engineering, a comprehensive understanding of its current development, applications, limitations, and open challenges remains elusive. Specifically, there is a lack of a holistic view of the practical usage of GenAI technology in software engineering scenarios. To address this gap, a literature review combined with focus groups was conducted over five months to develop a research agenda centered on GenAI for Software Engineering. This investigation identified 78 open Research Questions (RQs) across 11 areas of software engineering. The findings indicate that GenAI can facilitate partial automation and support decision-making throughout the software development lifecycle. However, the existing literature tends to focus primarily on areas such as software implementation, quality assurance, and software maintenance, leaving critical domains – such as requirements engineering, software design, and software engineering education – needing further research. Key considerations for implementing GenAI include industry-level assessments, dependability and accuracy, data accessibility, transparency, and sustainability aspects associated with the technology. While GenAI is poised to bring substantial changes to software engineering, the current state of research on this topic remains underdeveloped. This research agenda is intended to provide valuable insights and practical implications for both researchers and practitioners, informing them about existing applications while guiding future research directions in the field. Author in [11] discusses that Generative AI is considered a significant disruption in software development. Various platforms, repositories, clouds, and the automation of tools and processes have been shown to enhance productivity, reduce costs, and improve quality. With its rapidly expanding capabilities, Generative AI represents a substantial advancement in this domain. As a key enabling technology, it can serve multiple purposes, ranging from fostering creativity to automating repetitive and manual tasks. The capabilities of large language models (LLMs) further amplify the opportunities presented by Generative AI. However, this advancement also raises concerns regarding ethics, education, regulation, intellectual property, and potential criminal activities. The authors analyzed the potential of Generative AI and LLM technologies to shape future software development pathways. They propose four primary scenarios, model trajectories for transitions between them, and reflect on their relevance to software development operations. The motivation for this research is evident: the software development industry requires new tools to comprehend the potential, limitations, and risks associated with Generative AI, along with guidelines for its effective use. Author in [12] investigates the integration of Generative AI into software development education. The authors provide examples of formative and summative assessments that examine various aspects of ChatGPT, including its coding capabilities, its effectiveness in constructing arguments, and the ethical considerations surrounding the use of ChatGPT and similar tools in educational and workplace settings. The research is informed by survey insights indicating that learners in the Degree Apprenticeship Programme are highly interested in understanding and leveraging emerging AI technologies. Additionally, industrial partners express a strong desire for their employees to be adequately prepared to utilize Generative AI in their software engineering roles. To address this need, the authors propose embedding GenAI tools into the curriculum in a thoughtful and innovative manner. By developing assessments that encourage learners to critically evaluate AI-generated outputs, educators can enhance students' comprehension of the subject matter while mitigating the risk of AI tools performing the work for them.

3. Service-Oriented Architecture

Researchers have examined SOA from multiple perspectives, encompassing technology, business, and architectural viewpoints, which has led to a lack of a universally accepted definition. SOA is not a standalone technology, a specific product, or a simple solution to IT complexities, nor does it guarantee the resolution of all challenges within IT or information systems. Nevertheless, SOA is widely recognized as a conceptual framework with significant potential applications across business, IT, information systems, and enterprise-wide environments. [13]. Table 1 provides various descriptions of

SOA as outlined in previous studies.

#	Reference	Definition
1	[14]	Service-Oriented Architecture is a design approach for business environments that promotes the development of loosely coupled, interoperable, and technology-independent business services, enhancing functionality and flexibility.
2	[15]	Service-Oriented Architecture (SOA) is an architectural approach that fosters service-orientation, a perspective focused on services, service-driven development, and the results achieved through services.
3	[16]	Service-Oriented Architecture (SOA) represents a form of technological architecture that embodies the principles of service-orientation. When implemented through a Web services technology platform, SOA has the capacity to advance and uphold these principles across the domains of business processes and enterprise automation.
4	[17]	Service-Oriented Architecture (SOA) is a software architecture grounded in core components, including an application front-end, services, a service repository, and a service bus. Each service is composed of a contract, one or more interfaces, and an underlying implementation.
5	[18]	Service-Oriented Architecture (SOA) is a software architecture that begins with defining interfaces and structures the application as a network of interfaces, their implementations, and interrelated interface calls. SOA establishes a relationship between services and service consumers, where each software module is substantial enough to encapsulate a complete business function.

Table 1. SOA definition

The various definitions of SOA presented in Table 1 reflect distinct perspectives. While none are incorrect, there is general consensus among scholars that SOA can be understood as an architectural concept emphasizing loose coupling, reusability, interoperability, agility, and efficiency [1]. SOA focuses on decomposing business processes into discrete tasks and functions, often conceptualized as services. Numerous scholars have indicated that the integration of Service-Oriented Architecture (SOA) with complementary technologies can yield enhanced benefits for organizations [5]. The convergence of SOA and AI enhances system performance and responsiveness. By embedding AI capabilities into SOA frameworks, organizations can create intelligent services that learn and adapt over time [19]. For example, customer service applications can integrate AI-driven chatbots that improve responses based on user interactions [20]. This integration allows for real-time data processing and decision-making, significantly boosting customer experience and operational efficiency. Additionally, SOA enables the deployment of AI services across various platforms [21]. Organizations can leverage cloud-based AI services and APIs, allowing them to scale AI capabilities without extensive system reconfiguration. This flexibility helps businesses experiment with innovations like predictive analytics and automated decision-making without overhauling their entire IT infrastructure. Aligning these technologies will be crucial for driving efficiency and enhancing service delivery [22].

4. Generative AI

Generative artificial intelligence (AI) has rapidly gained prominence, particularly following the emergence of models such as ChatGPT and GPT-4, alongside similar advancements from competing organizations [23]. Beyond the ethical and practical concerns raised by scholars, it is evident that generative AI has found substantial applications in various sectors, rendering discussions about its potential impact increasingly relevant. The question is no longer whether generative AI will be influential, but rather to what extent it will shape societal dynamics and what potential harms may arise from its use in generating text and other forms of content [24]. Technological advancements inevitably lead to societal transformations, prompting critical inquiries into how new technologies influence, foster, or possibly undermine the concept of the “good society.” In this context, generative AI represents a significant instance of politically and culturally disruptive autonomous technology [25].

Research and development in generative AI (GAI) are focused on creating better, faster, and more capable models. However, the foundational principles, applications, and socio-economic impacts of GAI have not been thoroughly explored in academic discussions [26]. While GAI presents opportunities for innovation across various sectors, such as networked businesses and digital platforms, it also poses challenges, including issues related to transparency, biases, and potential misuse, that must be addressed for effective implementation [27]. Despite the importance of understanding key concepts, comprehensive examinations of generative AI remain lacking, resulting in an unclear understanding of its principles [28].

5. Methodology

In this section, we outline the methodology employed to test the integration of Generative AI into a Service-Oriented Architecture (SOA) for local web-based service discovery. **Overview of the SOC Environment:** A typical Service-Oriented Computing (SOC) setup consists of various components, including service directories, individual services, and communication protocols such as HTTP. Services are modular units that perform specific tasks, while the service directory acts as a registry for these services, allowing users to discover and access them efficiently. **Generative AI Model for Service Matching:** The generative AI model utilized for this study employs natural language processing techniques to recommend services based on user queries. In our approach, we trained the model using a dataset comprising common user requests and corresponding service descriptions. Alternatively, for simpler implementations, keyword detection techniques may be used, where the model matches user queries to service descriptions by identifying relevant keywords. For example, a user query like "Analyze customer data" would be matched to a "data analysis" service through keyword detection. **Local Web Interface:** To facilitate user interaction, we implemented a web interface using Flask. This framework allows users to submit their queries and receive service recommendations seamlessly. The local web interface serves as the front end, enabling users to enter their requests and view matched services based on the generative AI model's recommendations.

6. System Design and Architecture

The system is designed to enhance the traditional Service-Oriented Architecture by integrating Generative AI for improved service discovery.

Architecture Overview: The architecture consists of three main components: the service directory, the generative AI service, and the web interface.

1. **Service Directory:** This component stores a registry of services, including their descriptions and metadata. It allows for efficient querying and retrieval of available services based on user input.
2. **Generative AI Service:** This service acts as the core engine for matching user queries to relevant services. By employing natural language processing and machine learning techniques, it analyzes user input and recommends suitable services. The training data includes a variety of service descriptions and user queries to improve the model's accuracy and relevance.
3. **Web Interface:** Built with Flask, the web interface serves as the user entry point. Users can submit queries through a simple form, and the interface communicates with the generative AI service to retrieve service recommendations. The results are displayed in an intuitive format, enabling users to easily navigate the available services.

Communication Protocols: The system utilizes HTTP as the primary communication protocol between the web interface and the generative AI service. This standard protocol ensures that requests and responses are transmitted efficiently and securely, providing a smooth user experience. This design not only enhances service discovery through intelligent matching but also maintains the modularity and scalability inherent in Service-Oriented Architectures.

7. Implementation

This section outlines the implementation of a web-based service discovery application developed using

Flask, which demonstrates enhancements in service-oriented architectures (SOA) through the integration of generative AI techniques. The application is built using the Flask framework, a micro web framework in Python that promotes modular and scalable web application development. The architecture follows service-oriented principles, allowing for independent service modules that can be easily integrated or modified. The primary functionality revolves around matching user queries with relevant services. The `match_service` function, imported from `ai_model.py`, implements the logic for this matching process. When a user submits a query through the web interface, the application captures the input and passes it to the `match_service` function, which analyzes the query and identifies corresponding services. The application provides a simple user interface, consisting of an input field for users to enter their queries. Upon submission, the query is processed, and the matched services are returned and displayed on the results page (`results.html`). This interaction exemplifies a user-centric design in service-oriented architectures, where user feedback directly informs service offerings. While the current implementation utilizes a basic matching algorithm, there is significant potential for enhancing this functionality through generative AI techniques. Future iterations could involve integrating natural language processing (NLP) models to better understand user intent and generate more accurate service recommendations. This approach aligns with the concept of generative AI, which focuses on creating new content or suggestions based on learned patterns from existing data. The application has been tested to ensure proper functionality, with various user queries evaluated to assess the accuracy of the service matching. Initial results indicate that the application effectively returns relevant services based on user input. However, further testing is planned to refine the matching algorithm and explore the implementation of generative AI techniques for enhanced user experience and service relevance.

8. Discussion

In this section, we discuss the practical implementation and user interface of our web-based service discovery application, built to enhance service-oriented architectures using Generative AI. The application allows users to search for services via a simple query input, leveraging an AI model to retrieve and display relevant service matches. Screenshots of key interfaces are provided to illustrate the workflow, showcasing how the application processes user inputs and presents results. This interface design aims to create a smooth, user-friendly experience, aligning with the project's objective to simplify service discovery and make service-oriented computing more accessible for end-users.



Figure 1: Homepage

Figure 1 displays the application's home page, where users can enter queries to search for specific services. Upon entering a query, users can initiate the search by clicking the "Match Service" button, activating the backend service-matching functionality.

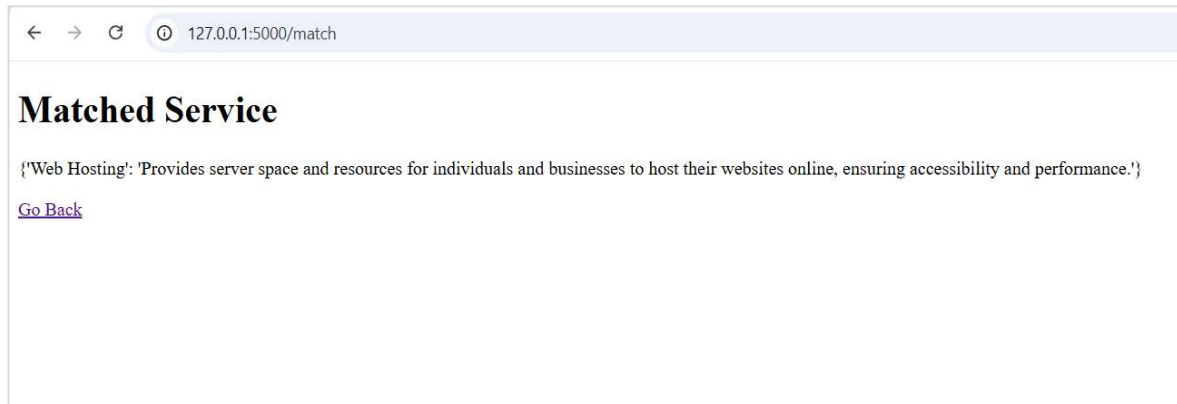


Figure 2: Result page

Figure 2 displays the results page, where the application shows services that match the user's query. This output provides users with descriptions and details of services that best align with their needs, thereby enhancing the service discovery process. The web application, built with Flask, facilitates user interaction by capturing search queries and providing AI-driven service recommendations. When a user submits a query through the form on index.html, the application sends the input to the /match route. Here, the match_service function processes the query using AI to find services that best match the user's needs. The resulting service suggestions are then displayed on the results.html page. This structure supports a seamless and dynamic user experience in service-oriented computing, where the integration of AI simplifies the process of discovering and matching relevant services based on specific user needs.

```
1. from flask import Flask, render_template, request
2. from ai_model import match_service # Import the service matching function
3.
4. app = Flask(__name__)
5.
6. @app.route('/')
7. def index():
8.     return render_template('index.html') # Render the main page
9.
10. @app.route('/match', methods=['POST'])
11. def match():
12.     user_query = request.form['query'] # Retrieve the user query from the form
13.     matched_service = match_service(user_query) # Call the matching function
14.     return render_template('results.html', service=matched_service) # Render results page
15.
16. if __name__ == '__main__':
17.     app.run(debug=True) # Run the application in debug mode
```

The match_service function is designed to intelligently match a user's input query with available services by examining keywords. It returns the relevant services and their descriptions if they align with the user's query or displays a message if no match is found.

```
def match_service(user_query): matched_services = {service: description for service, description in services.items()
if any(word.lower() in user_query.lower() for word in service.lower().split())} return matched_services if
matched_services else "No services found"
```

Input Parameter:

- user_query: This parameter takes the text input from the user (entered through a form), which specifies what they are looking for.

Service Matching:

- The function iterates over each service and description in the services dictionary. The services dictionary is expected to contain predefined service names as keys and their descriptions as values.

Keyword Matching:

- For each service, the code checks if any word in the service name matches a keyword from the user_query.
- The any() function is used to return True as soon as it finds a matching keyword in the user query.
- The lower() method ensures the matching process is case-insensitive, allowing words like "Database" and "database" to be considered equal.

Return Value:

- If any services match the query, the function returns a dictionary containing those matching services and their descriptions.
- If no matches are found, it returns a message: "No services found".

9. Conclusion

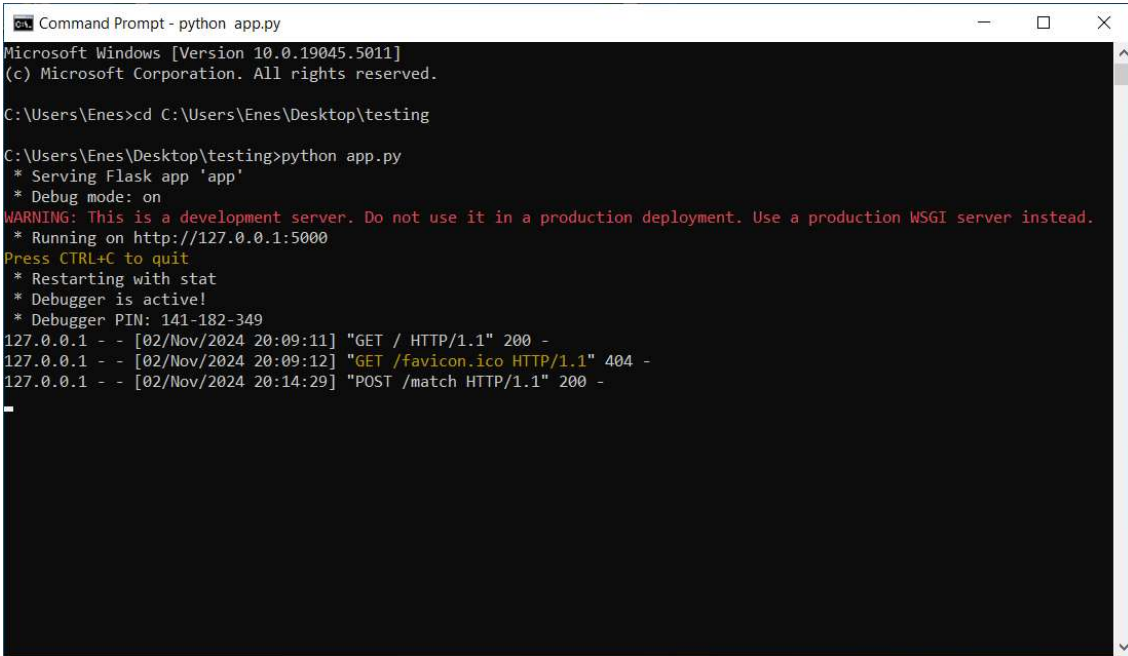
The present work underlines the importance of embedding Generative AI into a Service-Oriented Architecture for more effective and accessible service discovery. We implemented a local web-based application of service discovery where the underlying AI-powered matching model illustrates how Generative AI makes SOA frameworks smarter for better service relevance and more impactful user interaction. These results have shown that this approach supports smooth service retrieval, scalability, and adaptability pertinent to the SOA environment. Further developments could create even more advanced generative AI models in the future, applying deep learning to even higher levels of NLP to continue improving the user experience. This will enable the organizations to extend their capabilities within the SOA toward intelligent, responsive, and scalable systems that meet user needs with greater precision.

References

- [1] Naghmeh Niknejad, Waidah Ismail, Imran Ghani, Behzad Nazari, Mahadi Bahari, Ab Razak Bin Che Hussin,, "Understanding Service-Oriented Architecture (SOA): A systematic literature review and directions for further investigation," *Information Systems*, vol. 91, 2020.
- [2] Guillermo Rodríguez , J. Andrés Díaz-Pace , Álvaro Soria, "A case-based reasoning approach to reuse quality-driven designs in service-oriented architectures," *Information Systems*, vol. 77, pp. 167-189, 2018.
- [3] Georgios Katsikogiannis , Dimitrios Kallergis , Zacharenia Garofalaki , Sarandis Mitropoulos , Christos Douligeris, "A policy-aware Service Oriented Architecture for secure machine-to-machine communications," *Ad Hoc Networks*, vol. 80, pp. 70-80, 2018.
- [4] P. Gupta, T.P. Mokal, D.D. Shah, K.V.V. Satyanarayana, "Event-driven SOA-based iot architecture," *International Conference on Intelligent Computing and Applications. Advances in Intelligent Systems and Computing*, Springer, Singapore, pp. 247-258, 2018.
- [5] M.H.I. Hamzah, F. Baharom, H. Mohd, "An exploratory study for investigating the issues and current practices of service-oriented architecture," *J. Inf. Commun. Technol.*, vol. 18, no. 3, p. 273-304, 2019.
- [6] T. Hayet, J. Knani, "SOAP-based web service for localization of multi-robot system in cloud," *Advances in Intelligent Systems and Computing*, vol. 857, pp. 398-410, 2019.

- [7] Gaurav Raut, Apoorv Singh, "Generative AI in Vision: A Survey on Models, Metrics and Applications," *Computer Vision and Pattern Recognition*, pp. 1-12, 2024.
- [8] Prafulla Dhariwal, Alex Nichol, "Diffusion Models Beat GANs on Image Synthesis," *Machine Learning*, pp. 1-44, 2021.
- [9] G. Rodríguez, J.A. Díaz-Pace, Á. Soria, "A case-based reasoning approach to reuse quality-driven designs in service-oriented architectures," *Inf. Syst.*, pp. 167-189, 2018.
- [10] Anh Nguyen-Duc, Beatriz Cabrero-Daniel, Adam Przybylek, Chetan Arora, Dron Khanna, Tomas Herda, Usman Rafiq, Jorge Melegati, Eduardo Guerra, Kai-Kristian Kemell, Mika Saari, Zheyang Zhang, Huy Le, Tho Quan, Pekka Abrahamsson, "Generative Artificial Intelligence for Software Engineering -- A Research Agenda," *Software Engineering*, pp. 1-87, 2023.
- [11] Jaakko Sauvola, Sasu Tarkoma, Mika Klemettinen, Jukka Riekkki and David Doermann, "Future of software development with generative AI," *Automated Software Engineering*, vol. 31, 2024.
- [12] Olga Petrovska, Lee Clift, Faron Moller, Rebecca Pearsall, "Incorporating Generative AI into Software Development Education," *CEP '24: Proceedings of the 8th Conference on Computing Education Practice*, pp. 37 - 40, 2024.
- [13] Eric A. Marks, Michael Bell, *Service-Oriented Architecture: A Planning and Implementation Guide for Business and Technology*, Wiley - Online Library, 2012.
- [14] Sean Wheller and SYSPRO (PTY) Ltd, *A Guide to Service Oriented Architecture*, 2006.
- [15] V. Haren, *SOA Source Book*, Van Haren Publishing, Zaltbommel, www.vanharen.net, 2009.
- [16] Thomas Erl, "Sample Chapter 16 from "Service-Oriented Architecture: Concepts, Technology, and Design"," Donnelley in Crawfordsville, Indiana, 2005.
- [17] D Krafzig, K Banke, Dirk Slama, *Enterprise SOA - Service-Oriented Architecture Best Practices*, 2003.
- [18] T. G. K. Vasista and Mohammed A. T. AlSudairi, "Service-Oriented Architecture (SOA) and Semantic Web Services for Web Portal Integration," in *Proceedings of the Second International Conference on Advances in Computing and Information Technology (ACITY) July 13-15, Chennai, India - Volume 2*, 2013.
- [19] Erl, T., "Service-Oriented Architecture: Concepts, Technology, and Design," in *Prentice Hall*, 2005.
- [20] García-Magariño, I., Pacheco, J. A., & Pérez-Sanagustín, M., "Cloud-based AI services in a service-oriented architecture," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 9, no. 1, 2020.
- [21] Peters, K. , "Leveraging AI in customer service through SOA," *International Journal of Information Systems and Project Management*, vol. 6, no. 1, pp. 23-24, 2018.
- [22] Russell, S., & Norvig, P., *Artificial Intelligence: A Modern Approach*, Prentice Hall, 2010.
- [23] Henrik Skaug Sætra , "Generative AI: Here to stay, but for good?," *Technology in Society (ScienceDirect)*, pp. 1-5, 2023.
- [24] Charla Griffy-Brown , Brian D. Earp , Omar Rosas, "Technology and the good society," *Technology in Society (ScienceDirect)*, vol. 53, pp. 1-3, 2018.
- [25] L. Winner, *Autonomous technology: Technics-out-of-control as a theme in political thought*, 1978.
- [26] Strobel, G., Schoormann, T., Banh, L., & Möller, F. , "Artificial Intelligence for Sign Language Translation – A Design Science Research Study," *ommunications of the Association for Information Systems*, vol. 53, pp. 42-64, 2023.
- [27] Strobel, G., Banh, L., Möller, F., & Schoormann, T, "Exploring generative artificial intelligence: A taxonomy and types," *Hawaii International Conference on System Sciences (HICSS 2024), Hawaii, USA.*, 2024 .
- [28] Sara Moussawi, Marios Koufaris , Raquel Benbunan-Fich , "How perceptions of intelligence and anthropomorphism affect adoption of personal intelligent agents," *Electronic Markets* , vol. 31, p. 343–364, 2020.

Appendix



```
Command Prompt - python app.py
Microsoft Windows [Version 10.0.19045.5011]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Enes>cd C:\Users\Enes\Desktop\testing

C:\Users\Enes\Desktop\testing>python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 141-182-349
127.0.0.1 - - [02/Nov/2024 20:09:11] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [02/Nov/2024 20:09:12] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [02/Nov/2024 20:14:29] "POST /match HTTP/1.1" 200 -
```

The server log provides a detailed view of the Flask application's runtime environment, including initialization, requests, and responses. Initially, the command `cd C:\Users\Enes\Desktop\testing` navigates to the application directory, followed by `python app.py`, which starts the Flask application. Once initiated, the server confirms the main module (`app`) is in debug mode, indicated by `* Debug mode: on`, which allows for error tracing and auto-reloading on code changes. A development server warning appears, reminding that this setup is not suitable for production, suggesting a WSGI server for deployment instead. The server then runs locally at `http://127.0.0.1:5000`, where users can access the application. Additionally, Flask's `stat` mechanism restarts the server when it detects file changes, activating the debugger and providing a secure PIN (`Debugger PIN: 141-182-349`) for accessing debug features. The log captures three HTTP requests, each indicating the method, endpoint, status code, and timestamp. First, a `GET /` request (status 200) for the homepage completes successfully, followed by a `GET /favicon.ico` request with a 404 status, indicating that the favicon was not found. Finally, a `POST /match` request logs a user query submission to the match endpoint, which completes successfully with a 200 status. This log outlines the initialization, interaction, and response sequence of the application, providing a complete snapshot of the development environment and typical request handling.