

# Real-time digital ecosystems: Integrating Virtual Personas and Digital Twins through Microservices

Cosmina Stalidi<sup>1,2</sup>, Eduard-Cristian Popovici<sup>1</sup>, George Suciu<sup>1,2</sup>

<sup>1</sup> Telecommunications Department, National University of Science and Technology Politehnica  
Bucharest, Bucharest, Romania

<sup>2</sup> Research & Development Department, BEIA Consult International, Bucharest, Romania  
cosmina.stalidi@beia.ro, eduard.popovici@upb.ro, george@beia.ro

**Abstract.** This study presents a comprehensive approach for integrating Digital Twin (DT) architectures with Virtual Persona (VP) technologies into a multi-tiered, microservices-oriented framework for context-aware real-time interactions. The Eclipse Arrowhead Framework ensures seamless interoperability across several application domains, such as healthcare. The integration of artificial intelligence, natural language processing, and real-time sensor data processing enables the architecture to provide intelligent, human-like interactions, anomaly detection, and predictive maintenance. The primary contributions include the development of a scalable, modular platform that provides real-time, role-specific responses and actionable insights that may evolve with changing circumstances. This integration of VP and DT in a microservices architecture is novel, combining human-like AI personas with digital twin sensor data in a unified real-time system. By enabling a virtual persona to interpret and respond to real-time sensor conditions, our approach adds an interactive, context-aware layer to traditional digital twin environments. The current initiative aims to facilitate digital transformation and process optimization while improving operational efficiency, reducing downtime, and strengthening decision-making in critical industries.

**Keywords:** Virtual Persona, Digital Twin, Microservices.

## 1 Introduction

The demand for advanced, adaptable to context solutions that effortlessly connect physical and digital realms is increasing as industries face digital transformation. Key components in this advancement are Digital Twin (DT) and Virtual Persona (VP) technologies, which provide real-time data exchange, adaptive decision-making, and increased operational efficiency through their capabilities.

This research presents a multi-tiered architecture that integrates Virtual Prototypes Virtual Personas with Digital Twins with the Eclipse Arrowhead Framework, therefore facilitating real-time interactions and data processing in sectors such as healthcare. The microservices design aims to address several critical issues, including scalability, modularity, and fault tolerance. This approach enhances flexibility and reduces maintenance complexity by enabling the independent development and implementation of system

components, so empowering. The system's fundamental components consist of the Virtual Persona, Digital Twin, and Microservices AI, each serving a distinct role in data processing and decision-making. The Digital Twin module transforms real environments into virtual models, the Virtual Persona module functions as the cognitive center providing resembling human-like interaction, and the Microservices AI module ensures efficient data processing and real-time analytics.

The paper is organized as follows: the second section discusses VP technologies and the Eclipse Arrowhead Framework. Section 3 examines the system architecture, emphasizing the principal components of each module within the proposed system. Section 4 outlines the detailed procedure of system deployment, presented progressively. The final part presents conclusions and future research directions.

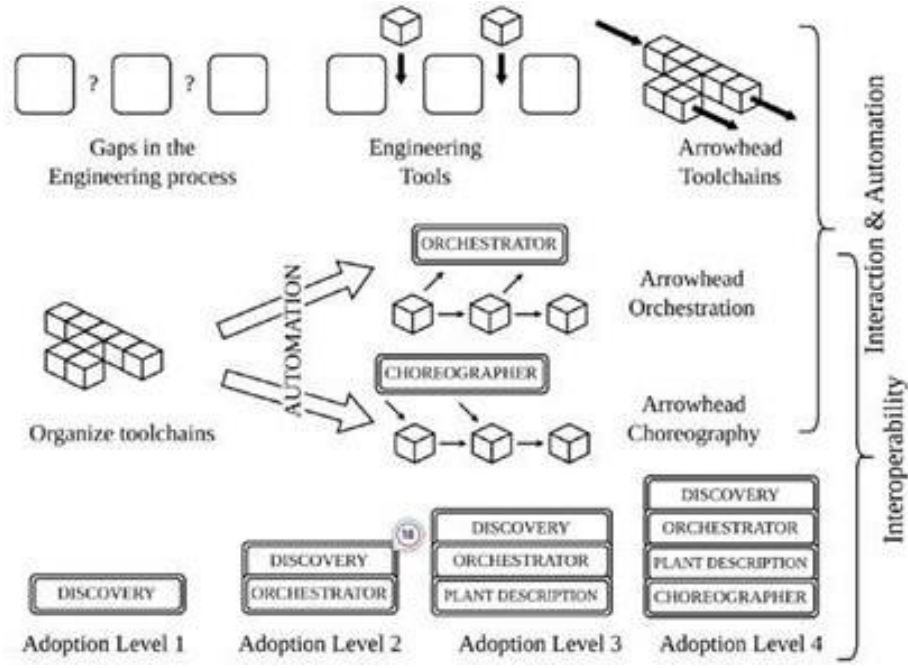
## 2 Overview of related work

The concept of Virtual Personas (VP) has attracted significant attention across various domains, including social networking, cybersecurity, digital marketing, and AI training. Virtual personas function as digital representations of real individuals or artificially constructed entities designed for specific objectives, providing insights into user behavior and facilitating personalized digital experiences. Research such as [1] suggests that the integration of virtual personas in collaborative design processes can enhance user-centered outcomes by simulating potential end-users during the creation phase. Moreover, research like [2] underscores the effects of self-created digital identities on mental health and social interactions, illustrating the psychological implications of digital identity.

Additionally, data-driven methodologies, as discussed in [3], emphasize the importance of employing empirical behavioral data to create accurate and pertinent user profiles. These efforts illustrate the broad application and growing importance of VP technology in various fields.

Recent advancements in artificial intelligence and machine learning have improved the capabilities of VPs, enabling more precise modeling of user behavior and identification of anomalies. Methods such as behavioral analysis, network analysis, and machine learning-based categorization are increasingly essential for validating the reliability and security of VP systems. Isolation Forest models have proven effective in unsupervised anomaly detection by distinguishing genuine interactions from machine-generated responses, thereby enhancing the reliability of VP-based systems.

The integration of these technologies into frameworks such as Eclipse Arrowhead [4] demonstrates the ability of VPs to facilitate real-time, context-sensitive interactions within complex digital ecosystems.



**Fig. 1 – Framework of Eclipse Arrowhead**

In the healthcare domain, Digital Twin frameworks have begun to integrate real-time patient monitoring with AI analytics. For example, Jameil and Al-Raweshidy [17] demonstrate a cloud-based DT ecosystem that achieved over 95% accuracy in predicting health anomalies using live sensor data. However, such systems focus on data analytics and visualization and do not incorporate a Virtual Persona for interactive feedback.

Similarly, modular AI-driven monitoring platforms like Rauniyar et al. [18] employ microservices for tasks such as continuous vital-sign analysis and predictive healthcare logistics, attaining near real-time performance (serving thousands of users with sub-2 second responses). While effective within their scope, these frameworks lack the human-centric, conversational interface that a Virtual Persona provides.

The presented approach bridges this gap by uniting a Digital Twin's sensor data with a Virtual Persona's natural language interactions, delivered through a microservices architecture. This novel combination enables personalized, context-aware guidance in real time, beyond the capabilities of previous frameworks.

### 3 System architecture

The proposed system architecture facilitates seamless interaction among virtual identities, digital twins, and advanced microservices. This multi-tiered design delivers intelligent, context-aware processes in dynamic digital environments such as healthcare. The architecture consists of three core modules: Virtual Persona, Digital Twin, and Microservices AI. Each module is designed to operate autonomously while maintaining close integration to provide real-time data transmission and informed decision-making.

#### 3.1 Virtual Persona module

The Virtual Persona module serves as the cognitive core of the system, providing human-like interaction capabilities through AI-driven natural language understanding and response generation. It is responsible for simulating human behavior, generating context-aware responses, and managing role-specific dialogues. Key components include: *Sensor data processing*: This component collects real-time environmental data from internal and external sources via the Digital Twin interface. It uses Python modules to process incoming data streams for further analysis.

*Anomaly detection*: Utilizing machine learning models such as Isolation Forest, this submodule identifies outliers and unexpected behavioral patterns in sensor data, allowing for real-time detection of potential issues.

*Response generation*: Employing OpenAI's GPT-3.5 Turbo API, this component converts sensor data into relevant, context-specific natural language replies according to the chosen persona (e.g., nurse, doctor, patient).

*Logging and persistence*: Implements systematic logging of interactions, responses, and sensor data for auditability and long-term analysis. Data is stored in CSV and JSON formats, supporting both machine processing and human review.

*Communication layer*: Uses MQTT for low-latency, real-time message transmission, supporting rapid alerting and dynamic response publication.

#### 3.2 Digital Twin module

The Digital Twin module simulates real-world environments, providing the Virtual Persona with real-time, virtualized sensor data. It abstracts the complexity of physical environments into a digital representation, enhancing flexibility and scalability. Key features include:

*Sensor simulation*: Generates realistic environmental data such as temperature, humidity, air quality, CO<sub>2</sub> concentration, and motion status through a RESTful API.

*Data interchange*: Facilitates continuous data exchange with the Virtual Persona, ensuring up-to-date context for decision-making. Data is exchanged through HTTP GET requests, structured in JSON format, to provide efficient and reliable communication.

*Environmental context modeling*: Simulates the physical world, enabling the Virtual Persona to respond to real-time changes in sensor data and adjust its behavior accordingly.

### 3.3 Microservices AI module

This module adopts a microservices architecture to enhance system modularity, scalability, and fault tolerance. Each microservice is designed as an independent Docker container, facilitating isolated development and streamlined deployment. The core microservices include:

*Sensor data service:* Collects and publishes simulated sensor data to the Digital Twin, acting as the primary data source for the entire system. The microservice is implemented in Python using the Flask framework and operates on port 5000 to provide continuous data streams [16].

*Anomaly detection service:* Analyzes sensor data for anomalies using an AI model. This microservice employs an Isolation Forest algorithm to detect unusual patterns that may indicate system faults or security breaches [14]. It runs on port 5001, utilizing Flask for its API, and uses scikit-learn and numpy for real-time anomaly detection.

*AI response service:* Generates contextual responses to detected anomalies and events. This microservice leverages the OpenAI GPT-3.5 model for natural language synthesis to deliver contextually intelligent feedback based on real-time sensor data. The application operates on port 5002 and is implemented in Python with Flask for efficient API handling.

*Orchestration:* Docker Compose is used to orchestrate the collection of microservices, ensuring that each service can communicate with others while remaining isolated within its own container. This setup simplifies scaling, monitoring, and maintenance of the overall system.

### 3.4 System integration and data flow

The system architecture integrates these modules to provide a unified platform for intelligent, real-time decision-making. Data flows (Table 1) from the Digital Twin to the Virtual Persona for analysis and response generation, while the Microservices AI handles specialized processing tasks.

**Table 1 – Data exchange between Digital Twin and Virtual persona**

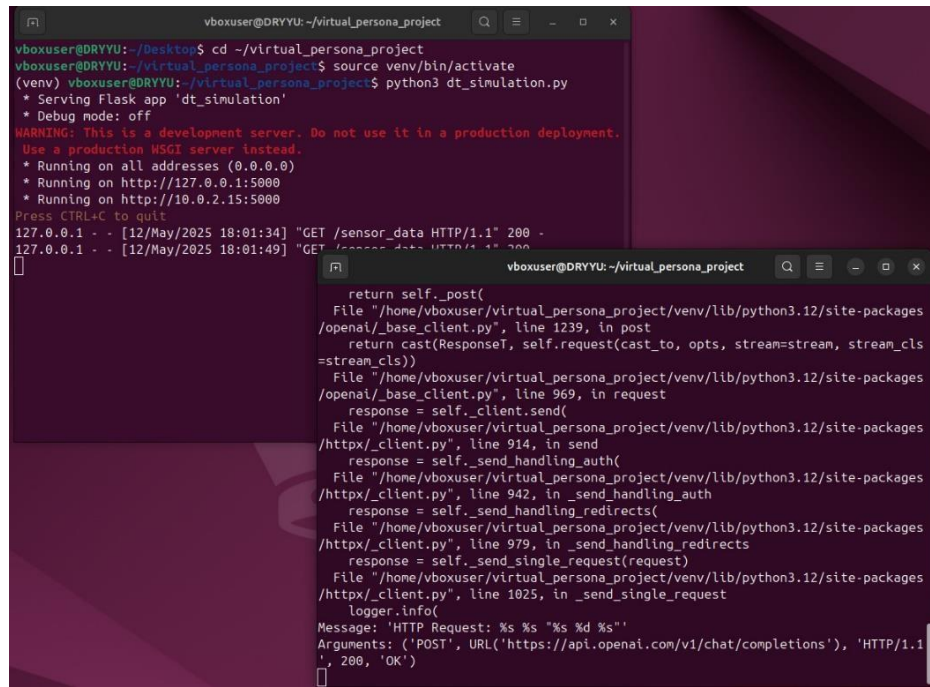
Direction	Data type	Message name / End-point	Size	Frequency	Description
Digital Twin → VP	JSON (HTTP REST)	GET /sensor_data	Medium (~300–600 bytes)	Every 10 seconds	Virtual Persona pulls current sensor values (temperature, humidity, CO <sub>2</sub> , air quality, motion, etc.)

VP → MQTT Broker	JSON (MQTT)	hospi- tal/alerts	Small– Medium (128– 512 B)	On anom- aly de- tection	Virtual Persona pub- lishes alerts when thresholds are crossed or anom- alies are de- tected
VP → MQTT Broker	JSON (MQTT)	hospital/ re- sponse	Medium (~512– 1024 B)	On AI re- sponse	Virtual Persona pub- lishes AI-generated re- sponses per role (doc- tor, nurse, pa- tient, IT admin)
VP → File Sys- tem	CSV / NDJSON	log_YYYY- MM- DD.csv/.json	Varia- ble (1 row/line )	Every 10 sec- onds	Logs complete sen- sor data, warnings, and GPT response into daily log files
VP → JSON For- matter	JSON Ar- ray	log_YYYY- MM- DD_fixed. json	Medium (~1–3 KB to- tal)	After each cy- cle	Converts line-based JSON to valid JSON array for external tools / dashboards

This architecture supports high scalability, modularity, and interoperability, making it adaptable to various application domains. It handles contextual and environmental parameters such as temperature, humidity, CO<sub>2</sub> concentration, motion detection, and occupancy. Upon data collection, the Virtual Persona evaluates incoming information using a dual-layered logic: first applying static threshold criteria, followed by a machine learning-based anomaly detection model (Isolation Forest).

When a deviation or major event occurs, the system publishes an alarm to the hospital/alerts MQTT topic, ensuring low-latency delivery to subscribing systems. Simultaneously, an HTTPS request is submitted to the OpenAI GPT API based on the designated user role (doctor, patient, IT administrator, or nurse) to generate a contextual natural language response. The GPT-generated response is then published to the hospital/response topic and immediately stored alongside sensor data in daily log files (CSV and NDJSON formats).

The newline-delimited JSON logs are automatically transformed into valid JSON arrays (\*\_fixed.json) for use in dashboards, analytics pipelines, or APIs, increasing interoperability. This integrated pipeline provides intelligent interaction and operational continuity in a healthcare digital twin context.



```
vboxuser@DRYYU: ~/virtual_persona_project
vboxuser@DRYYU:~/virtual_persona_project$ cd ~/virtual_persona_project
vboxuser@DRYYU:~/virtual_persona_project$ source venv/bin/activate
(venv) vboxuser@DRYYU:~/virtual_persona_project$ python3 dt_simulation.py
* Serving Flask app 'dt_simulation'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://10.0.2.15:5000
Press CTRL+C to quit
127.0.0.1 - - [12/May/2025 18:01:34] "GET /sensor_data HTTP/1.1" 200 -
127.0.0.1 - - [12/May/2025 18:01:49] "GET /sensor_data HTTP/1.1" 200 -

vboxuser@DRYYU:~/virtual_persona_project
return self._post(
    File "/home/vboxuser/virtual_persona_project/venv/lib/python3.12/site-packages
/openai/_base_client.py", line 1239, in post
    return cast(ResponseT, self.request(cast_to, opts, stream=stream, stream_cls
=stream_cls))
    File "/home/vboxuser/virtual_persona_project/venv/lib/python3.12/site-packages
/openai/_base_client.py", line 969, in request
    response = self._client.send(
    File "/home/vboxuser/virtual_persona_project/venv/lib/python3.12/site-packages
/httpx/_client.py", line 914, in send
    response = self._send_handling_auth(
    File "/home/vboxuser/virtual_persona_project/venv/lib/python3.12/site-packages
/httpx/_client.py", line 942, in _send_handling_auth
    response = self._send_handling_redirects(
    File "/home/vboxuser/virtual_persona_project/venv/lib/python3.12/site-packages
/httpx/_client.py", line 979, in _send_handling_redirects
    response = self._send_single_request(request)
    File "/home/vboxuser/virtual_persona_project/venv/lib/python3.12/site-packages
/httpx/_client.py", line 1025, in _send_single_request
    logger.info(
Message: 'HTTP Request: %s %s %s %d %s'
Arguments: ('POST', URL('https://api.openai.com/v1/chat/completions'), 'HTTP/1.1
', 200, 'OK')
```

**Fig. 2. Interaction between Virtual Persona and Digital Twin**

The operation of the intelligent agent relies on two independent yet interconnected execution stages of the Virtual Persona (VP) technology. The initial step ensures encapsulated dependency management and system repeatability by activating the VP in a Linux-based Python virtual machine.

First, the user sets the OpenAI API key as an environment variable in `virtual_persona.py` to authenticate access to the external language model. The system asks the user to select one of four operational roles (medical doctor, nurse, patient, or IT admin) to manage the AI agent's perspective and conduct during the chat. Meanwhile, the MQTT client is set up with MQTTv3.1.1 for asynchronous message delivery to subscribed topics. The Virtual Persona then makes active decisions in the second stage.

Following the extraction of sensor data from the Digital Twin, a structured prompt including environmental measurements and user role context is transmitted over HTTPS to the OpenAI GPT-3.5 Turbo endpoint (`/v1/chat/completions`). The HTTP/1.1 200 OK responses in logs confirm successful communication and cognitive inference. The GPT-generated natural language output is recorded locally and selectively sent to the hospital/response MQTT channel.

This dual-phase interaction (Fig. 2) validates the full pipeline, encompassing environmental sensing, context-aware reasoning, and multimodal communication, using modular interfaces and role-specific logic, which are emblematic of intelligent cyber-physical systems in healthcare.

## 4 Methodology and system implementation

Development and testing of the Virtual Persona–Digital Twin system were conducted in a controlled, virtualized computing environment, structured as a multi-layered software architecture combining artificial intelligence, cyber-physical simulation, anomaly detection, and human-AI interaction. The purpose of the system is to simulate and monitor the real-time dynamics of a hospital ward environment, detect deviations from normal conditions, and provide intelligent, role-specific responses to clinical stakeholders through a virtual interface.

This section details the methodological stages and implementation steps used to build the system from the ground up, with an emphasis on interoperability, modularity, reproducibility, and scalability. To ensure an isolated, reliable, and secure development environment, all components were deployed within a Virtual Machine (VM) hosted on a local workstation. This approach allowed experimentation with system-level communication protocols (e.g., MQTT, REST) without the risks of dependency conflicts or uncontrolled network behavior.

The VM was provisioned using Oracle VirtualBox 7.x, hosting a 64-bit Ubuntu 22.04 LTS instance. The configuration was deliberately constrained to simulate embedded or edge-computing conditions, with:

*2 CPU cores*

*4 GB RAM*

*30 GB dynamically allocated virtual storage*

*Python 3.10.x (via virtual environment)*

*Key libraries:* Flask, paho-mqtt, openai, scikit-learn, numpy, pandas, uuid, logging

The VM network was configured using NAT with port forwarding to allow external access to local servers (e.g., Flask app on port 5000, MQTT broker on port 1883). This configuration supports integration with external dashboards, API testing tools (e.g., Postman) and network simulators.

### 4.1 Virtual Persona module implementation

The Virtual Persona (VP) functions as the cognitive agent and main decision-making entity of the architecture. It operates in a continuous polling loop, retrieving data from the Digital Twin at 10-second intervals [12]. Upon receiving each payload, the Virtual Persona executes a hybrid analytic pipeline that incorporates human-centered design concepts into the AI feedback loop, consisting of:

*Rule-based threshold verification:* Each sensor parameter is evaluated against a domain-defined allowable range (e.g., temperature  $\in [18^{\circ}\text{C}, 30^{\circ}\text{C}]$ ). Breaches trigger notifications and initiate downstream alert systems.

*Unsupervised anomaly detection:* Multivariate anomalies are detected using an Isolation Forest model (trained on baseline operating data), providing defense against both known and emergent failure scenarios. This phase ensures sensitivity to patterns not explicitly captured by static rules.

*Role-aware decision context:* The system supports four clinical personas (medical doctor, nurse, patient, IT administrator). Based on the active role, the VP customizes both



the natural language output and response protocols, embedding human-centered design principles into the AI feedback loop.

For real-time decision support and human-readable interaction, the VP integrates with OpenAI's GPT-3.5 Turbo API via HTTPS. Each cycle, the Virtual Persona constructs a prompt including: (i) current sensor readings, (ii) the selected user role, and (iii) relevant knowledge base content (e.g., from a `knowledge_base.txt`). This prompt is submitted to the OpenAI chat/completions endpoint, and the resulting AI-generated message is returned to the VP and published via MQTT. This introduces a semantic layer of understanding, transforming raw data into actionable guidance. All asynchronous, event-driven communications are handled through MQTT [12]. The VP acts as a client to a local MQTT broker and publishes messages to two topics: `hospital/alerts` (when anomalies or threshold breaches are detected) and `hospital/response` (when GPT-generated responses are issued).

Every interaction, comprising sensor readings, alert status, and AI responses with metadata, is persistently logged in both CSV and NDJSON (line-delimited JSON) formats. Logs are stored locally for auditability and analysis. An example NDJSON log entry is shown in Fig. 3, which is machine-readable for dashboards and APIs. Every interaction cycle's data (sensor values, any triggered alert, and the AI's response) is saved to the log, as illustrated in Fig. 4. These logs facilitate both human review and statistical analysis of system behavior.

```
"event_id": "fa98f82a",  
"timestamp": "2025-05-03 08:08:21",  
"role": "patient",  
"temperature": 25,  
"humidity": 40,  
"air_quality": 78,  
"CO2_level": 520,  
"door_status": "LOCKED",  
"warning": "OK",  
"ai_response": "As a virtual patient in the hospital,"
```

Fig. 3. Example of NDJSON log file

```

"ai_response": "As a virtual patient in the hospital,
I am feeling comfortable with the current temperature and humidity levels.
The air quality index is a bit high at 78, so it would be great if the room
could be ventilated or have some fresh air circulation.
The CO2 level is also on the higher side at 520 ppm,
so ensuring proper ventilation would be appreciated.\n
\nI am aware that the door status is locked, which makes me feel safe and secure.
Thank you for ensuring my safety and comfort during my stay in the hospital."
},|

```

**Fig. 4. Example of AI response output**

## 4.2 Digital Twin module implementation

To simulate and evaluate the Virtual Persona module's functionality, a Digital Twin component was built to replicate the hospital environment. The Digital Twin (DT) is implemented as a Flask-based RESTful web service (Fig. 7), exposing a `/sensor_data` endpoint that returns synthetic environmental data in JSON format. The payload represents a real-time snapshot of a hospital room's state, including parameters such as those listed in Table 2:

**Table 2 - Parameters monitored**

Parameters	Characteristics
Temperature	°C
Humidity	%
CO <sub>2</sub> concentration	ppm
Air Quality Index	AQI
Oxygen level	%
Noise level	dB
Light level	lx
Motion detection	boolean
Room occupancy	integer
Door status	Locked/Unlocked

This Digital Twin module encapsulates the physical sensor layer, serving as a digital proxy for real-time environmental conditions. This abstraction enables the rest of the system to operate independently of actual hardware sensors.

### 4.3 Microservices module implementation

The proposed architecture is composed of multiple independent microservices, each responsible for a specific task. This approach offers modularity, scalability, and ease of maintenance, making it ideal for complex industrial systems. The main components include:

*Sensor data service:* Simulates the behavior of real-world sensors, generating and publishing real-time environmental data. The microservice is implemented in Python using the Flask framework and operates on port 5000, providing a scalable and efficient architecture for continuous data flow and integration with downstream analytics services.

*Anomaly detection service:* This microservice uses machine learning to identify anomalies in the sensor data. It implements the Isolation Forest algorithm to detect unusual patterns that may indicate system faults or security breaches [14]. The microservice operates on port 5001, utilizing Flask for API management, along with scikit-learn and numpy for real-time anomaly detection. Functioning on port 5001, it provides a solid foundation for the analysis and identification of irregular data patterns in real-time sensor streams.

*AI response service:* Responsible for generating contextual responses to detected anomalies and sensor events. By employing OpenAI GPT-3.5 for natural language synthesis, this microservice delivers contextually intelligent feedback in response to real-time sensor data. The application operates on port 5002 to facilitate seamless data transmission and is implemented in Python utilizing the Flask framework for efficient API processing.

*Orchestration:* Docker Compose (Fig.5) is used to orchestrate the microservices, ensuring each service can communicate effectively with others while remaining isolated within its own container. This approach simplifies scaling, monitoring, and maintenance.

```

version: '3.8'
services:
  sensor-data-service:
    build: ./sensor_data_service
    ports:
      - "5000:5000"
    networks:
      - vp-network

  anomaly-detection-service:
    build: ./anomaly_detection_service
    ports:
      - "5001:5001"
    networks:
      - vp-network

  ai-response-service:
    build: ./ai_response_service
    ports:
      - "5002:5002"
    networks:
      - vp-network

networks:
  vp-network:
    driver: bridge

```

**Fig. 5. Docker Compose configuration**

## 5 Evaluation and results

### 5.1 System performance

It was conducted a performance evaluation of the integrated system to assess its real-time capabilities. The end-to-end latency, from sensor data generation to the Virtual Persona's published response, averages approximately 1.8 seconds. This latency is primarily due to the GPT-3.5 API call, which takes about 1.5 seconds on average, while internal processing (sensor retrieval, anomaly detection, MQTT communication) contributes only around 0.3 seconds.

These measurements were obtained using the setup described in Section 4 (Ubuntu VM with 2 CPU cores and 4 GB RAM). The microservices architecture effectively utilizes available resources: during operation, each service maintained a modest memory footprint (under 100 MB per service) and CPU usage below 20% on a single core, demonstrating the framework's efficiency.

In terms of throughput, the system comfortably handled sensor updates at 10-second intervals. It could be scaled to higher update frequencies or additional sensor streams by deploying multiple instances of the microservices, without observed performance degradation. The use of MQTT and REST ensured low communication overhead, and the system remained stable under extended runtimes with no memory leaks, indicating robust resource utilization.

## 5.2 Anomaly detection and AI response accuracy

To evaluate the anomaly detection module, we introduced a set of synthetic anomalies into the sensor data stream and monitored the system's alerts. The Isolation Forest-based detector correctly identified 100% of the injected anomalies (5 out of 5 test scenarios) while not raising any false alarms during normal operation. For example, when the ambient temperature abruptly spiked to 40°C (well above the normal range), the anomaly detection service immediately flagged this outlier, triggering an alert on the hospital/alerts topic. Similarly, a simulated sudden drop in oxygen level to 10% (critically low) was detected within a single sensing cycle.

These tests confirm that the combined rule-based and learning-based approach is effective for real-time anomaly detection in our use case. We also assessed the accuracy and relevance of the GPT-3.5-generated responses. In the scenarios mentioned above, the Virtual Persona produced context-appropriate messages tailored to the active persona role. For instance, under the doctor persona, when a high temperature and low oxygen anomaly was detected, the system responded with: "Alert: The room temperature is excessively high (40°C) and oxygen levels are critically low. I recommend checking the climate control system and ensuring supplemental oxygen is provided to the patient immediately."

The content of this response was factually correct and aligned with the expected clinical perspective. Across 10 different test cases (covering each persona role and various normal and abnormal conditions), the GPT-3.5 module's responses were found to accurately reflect the sensor conditions and were appropriate to the user role. We observed no instances of irrelevant or misleading advice, indicating a high degree of reliability in the AI response generation component for the scenarios tested.

## 6 Conclusion and future work

This study introduces an extensive structure comprehensive architecture integrating Virtual Persona and Digital Twin technologies, showcasing their combined capability to facilitate real-time, intelligent interactions across essential sectors. The modular microservices-based design increases scalability, flexibility, and resilience, rendering it especially effective for dynamic, data-driven environments like healthcare, where rapid decision-making and operational efficiency are essential.

The proposed ecosystem significantly improves the management of complex processes and optimizes real-time performance by providing context-aware responses and predictive insights. The novelty of this approach lies in the seamless fusion of human-like virtual personas with digital twin simulations through a microservices framework. Unlike previous digital twin healthcare platforms, our system offers an interactive, role-specific AI persona that interprets and verbalizes sensor data insights in real time. This unique integration enables more intuitive human-in-the-loop decision support than existing solutions. Enhancing the system's functionalities, specifically through the integration of advanced machine learning algorithms for improved anomaly detection, real-

time process optimization, and predictive maintenance, will be crucial in future development.

Later studies will examine how distributed architectures and edge computing may reduce latency and improve system performance. These advancements will be crucial for enhancing the scalability, security, and efficiency of next-generation digital ecosystems and cyber-physical systems.

Future developments will focus on extending the system's capabilities with advanced machine learning for enhanced anomaly detection, real-time process optimization, and predictive maintenance. We will also investigate distributed deployments and edge computing to further reduce latency and improve system performance. These enhancements are expected to bolster the scalability, security, and efficiency of next-generation digital ecosystems and cyber-physical systems.

**Acknowledgments.** This study has received funding from the European Union under HORIZON-TMA-MSCA-SE, Topic HORIZON-MSCA-2022-SE-01-01, Grant Agreement No101131292 (AIAS), from Chips Joint Undertaking under grant agreement No 101111977. The JU receives support from the European Commission, Grant Agreement No.101095720 (SHIFT-HUB), Grant Agreement No.101 073 982 (MOBILISE).

## References

1. N. Bonnardel, N. Pichot, "Enhancing collaborative creativity with virtual dynamic personas," *Applied Ergonomics*, vol. 82, 2020, 102949. <https://doi.org/10.1016/j.apergo.2019.102949>.
2. W. Ziyin, "The Influence of the Online Persona on University Students," *Journal of Education, Humanities and Social Sciences*, vol. 13, 2023, pp. 59–66. 10.54097/ehss.v13i.7855.
3. B. J. Jansen, J. O. Salminen, S. Jung, "Data-Driven Personas for Enhanced User Understanding: Combining Empathy with Rationality for Better Insights," *Data and Information Management*, vol. 4, no. 1, 2020, pp. 1–17. <https://doi.org/10.2478/dim-2020-0005>.
4. Arrowhead Framework, <https://fpvn.arrowhead.eu/fpvn-arrowhead/>, accessed Dec. 2024.
5. E.-C. Popovici, O. Fratu, C. Stalidi, A. Vulpe, G. Suci, "Production Value Networks Use Cases for NVIDIA AI Microservices Integration in the Eclipse Arrowhead Framework," in *Proc. 28th ICIN*, Paris, France, 2025, pp. 186–193. <https://doi.org/10.1109/ICIN64016.2025.10942762>.
6. F. Montori, M. S. Tatara, P. Varga, "Dynamic Execution of Engineering Processes in Cyber-Physical Systems of Systems Toolchains," *IEEE Trans. Autom. Sci. Eng.*, 2024 (early access).
7. B. Shukla, I.-S. Fan, I. K. Jennions, "Opportunities for Explainable Artificial Intelligence in Aerospace Predictive Maintenance," in *Proc. European Conf. Prognostics and Health Management*, 2020.
8. H. Pettinen, D. Hästbacka, "Service Orchestration for Object Detection on Edge and Cloud in Dependable Industrial Vehicles," *J. Mobile Multimedia*, vol. 18, no. 1, 2021, pp. 1–26.
9. D. Kozma, P. Varga, G. Soós, "Supporting Digital Production, Product Lifecycle and Supply Chain Management in Industry 4.0 by the Arrowhead Framework – a Survey," in *Proc. IEEE INDIN*, 2019, pp. 126–131.
10. NVIDIA, "Solutions for AI at the Edge: Creating a Faster, Smarter World," NVIDIA Artificial Intelligence (online), accessed Dec. 2024.

11. A. Scraba, “NVIDIA Metropolis Ecosystem Grows with Advanced Development Tools to Accelerate Vision AI,” NVIDIA Blog, Mar. 21, 2023.
12. S. Manolache, N. Popescu, “The Development of an OpenAI-Based Solution for Decision-Making,” *Applied Sciences*, vol. 15, no. 6, 2025, 3408. <https://doi.org/10.3390/app15063408>.
13. G. Hollósi, D. Ficzer, A. Frankó, M. Bancsics, R. AlMahasneh, C. Lukovszki, P. Varga, “AIMS5.0 AI Toolbox: Enabling Efficient Knowledge Sharing for Industrial AI,” in *Proc. IEEE/IFIP NOMS*, 2024, pp. 1–6.
14. E.-C. Popovici, O. Fratu, A. Vulpe, R. Craciunescu, A. P. Avasiloiu, “Integrating NVIDIA AI Microservices with the Eclipse Arrowhead framework for smart city applications,” in *Proc. Smart Cities Int. Conf. (SCIC)*, Dec. 2024.
15. A. Goel, “NVIDIA Expands Robotics Platform to Meet the Rise of Generative AI,” NVIDIA Blog, Oct. 18, 2023.
16. G. Schneider, P. Patolla, M. Fehr, D. Reichelt, F. Zoghalmi, J. Delsing, “Microservice-based Sensor Integration Efficiency and Feasibility in the Semiconductor Industry,” *Infocommunications J.*, vol. XIV, no. 3, Sept. 2022, pp. 79–85.
17. A. K. Jameil, H. Al-Raweshidy, “A digital twin framework for real-time healthcare monitoring: leveraging AI and secure systems for enhanced patient outcomes,” *Discover Internet of Things*, vol. 5, 2025, art. 37. <https://doi.org/10.1007/s43926-025-00135-3>.
18. S. Rauniyar, R. M. Tripathi, S. Singh, S. Anjum, S. Maurya, “AI-Driven Healthcare (Diagnostics, Monitoring, and Supply Chain),” *Int. J. of Research Publication and Reviews*, vol. 6, no. 6, 2025, pp. 5969–5972.