AVE Trends in Intelligent Computing Systems

Integral Smart Vehicle Automation Framework Using IoT, Modern Software, and Infrastructure

Brikelda Licaj*
Department of Software Engineering, Canadian Institute of Technology (CIT), Tirana, Albania.
brikelda.licaj@cit.edu.al.

Dimitrios A. Karras
Department of General, National and Kapodistrian University of Athens (NKUA), Athens, Greece.
And
Department of Computer Science and Engineering, EPOKA University, Tirana, Albania.
dakarras@uoa.gr

*Corresponding author

Abstract: Although the shipbuilding sector is expanding at a never-before-seen rate, ship owners and operators are eager to maximize their new boat investments so they can provide passengers with a unique experience. Technology makes this feasible by enabling operators to provide integrated services like cabin automation, check-in, and booking. The user can accomplish this while relaxing in their cabin. Over the past ten years, the cruise ship business has undergone significant change, and it faces significant obstacles in keeping up with market expansion and meeting the evolving needs of potential passengers. Starting with the plan and ending with the vessel's commissioning, the process of building an underused vessel might take a very long period. As a result, shipbuilders and administrators face the enormous task of ensuring that contemporary ships will be practical for their planning needs throughout their lengthy benefits. Every component of the vessel must be selected with great care to ensure not just optimal utility but also future validation of the innovation. This paper aims to present the design of such a complex system architecture with an analysis of its interacting components as well as with the analysis of its corresponding development framework and its implementation details.

Keywords: Security and Docker; Internet of Things; Spring Boot; Simulator and PostgreSQL; Controller of Pipelines; Smart Vehicles; Shipbuilding and Cruise Lines; Complex Systems.

Cite as: B. Licaj and D. A. Karras, "Integral Smart Vehicle Automation Framework Using IoT, Modern Software, and Infrastructure," *AVE Trends In Intelligent Computing Systems*, vol. 1, no. 3, pp. 128–141, 2024.

Journal Homepage: https://avepubs.com/user/journals/details/ATICS

Received on: 29/01/2024, **Revised on:** 19/03/2024, **Accepted on:** 15/05/2024, **Published on:** 01/09/2024

1. Introduction

Shipbuilding is a challenging industry, and cruise lines and yards have to work closely together to find the most effective methods to avoid delays in the construction of their ships. The manufacturing of cabins is constrained by a great variety of constraints, the most notable of which are the thickness of the partition walls and the need to bring down the total weight, amongst many others. Because of this, it is vital to establish close coordination between operators, builders, and suppliers in order to provide the most ideal solution feasible in terms of design, cost, and reliability [7]. This may be accomplished by developing tight coordination. It is a helpful strategy to prepare mock-up cabins to test goods in a controlled environment that will reflect the conditions in which the gadgets will really be put. This is done to verify that the selected alternative can be implemented [8]. The sample room will be essential for determining whether the installation process is straightforward and for determining whether sufficient space is provided to house all of the electrical components [9]. This is because more complex systems sometimes need extra installation space. When the partitions are just 30 millimetres thick, the depth of installation is usually a challenge [10]. Additionally, every layer of material added to the wall results in an increase in both cost and, maybe more crucially, the wall's overall weight [11].

The carbon footprint left by these marine giants is now the subject of intense scrutiny, and the reduction in weight has the potential to make a considerable contribution to the overall reduction in the use of fossil fuels. In the effort to bring down the overall ship weight, every component, such as the copper busbar that distributes electricity to the staterooms, is under close inspection [12]. However, to decrease power use and, consequently, reduce the cross-sectional area of the busbar or cables that are necessary to supply the cabins, certain power outlets may be turned off as part of the cabin automation system [13]. Even while the operators have no intention of preventing passengers from using the electronic gadgets of their choice while on board, they have no intention of doing so either [14]. This will lower the weight by an even greater amount while also saving money on the busbar and protection for the electrical circuit [15].

As soon as the issues with the construction have been fixed, it is time to shift attention to the user interfaces, which are going to be a significant part of the experience that the passengers will have [16]. When passengers board the ship, they will surely have a positive impression due to the ship's attractive interior; however, having a control system that is well-designed and easy to use will help ensure that they keep having a positive image [17]. Because a cabin automation system that is too sophisticated would distract from the overall experience of the customer, all controls and convenience charging stations must be developed with usability in mind [18].

The normal length of a trip aboard a cruise ship is around seven days (two days are spent in hotels!!). Therefore, several different options for operating the system need to be provided, ranging from high technology, such as the use of the visitors' mobile device, to the most fundamental option possible, which is a light switch [19]. The operators want their customers to be able to relax and enjoy their trip without being distracted by complicated control panels and systems that are more advanced than necessary [20]. The only things that the passengers should notice about the automated cabin are the perks, such as a soothing environment in which they can access all the controls with the push of a button and scenarios that can always establish the appropriate atmosphere [26]. Only by working as a team closely together and side by side can all design and development issues be addressed and the best and most effective solutions be discovered [27]. The challenges that were listed above are only a few of the problems that operators, shipbuilders, and suppliers face when beginning from scratch with a new vessel [28].

1.1. Scope of the Study and Related Research

This study is mostly based on the qualitative research method. The paper identifies and investigates how designing and developing smart vehicles involving automation and IoT technologies, as well as modern software development frameworks, would be possible and how it will impact vehicle usage and the facilitation of users' lives specifically [29]. After an introduction to the Internet of Things and all its applications, as well as an introduction to Automation principles and methods, an implementation is herein presented using Spring Boot, Angular, PostgreSQL, Docker, and Kubernetes software development frameworks and infrastructure [30].

In order to implement this type of automation, along with the standard networking functionalities, including security aspects, we have included different smart objects in the system design for different uses of automation, such as smart fan, smart window, smart door, smart light, smart garage door, fire alarm and different sensors [31]. All these objects are simulated in this project. The design of automation facilities and services in smart ships and vehicles integrates various technologies to enhance operational efficiency, security, and user experience [32]. This integration involves a multidisciplinary approach that combines IoT, DevOps practices, and advanced software frameworks, ensuring a cohesive and intelligent system [33].

1.1.1. Integration of Technologies

- **IoT and Smart Systems**: IoT enables real-time data collection and monitoring, facilitating smarter decision-making in ship operations [1].
- **DevOps and CI/CD Pipelines**: Utilizing Jenkins and Docker for continuous integration and deployment streamlines the development process, ensuring rapid updates and security patches [2].
- **Data Management**: PostgreSQL serves as a robust database solution for managing the vast amounts of data generated by various subsystems, enhancing data accessibility and analysis [3].

1.1.2. Security Considerations

- **Cybersecurity Measures**: Implementing security protocols is crucial to protect sensitive data and ensure the integrity of automated systems [34].

- **Collaborative Frameworks**: The integration of connected systems, such as Automated Guided Vehicles (AGVs), enhances operational safety and efficiency in port logistics [4].

While the focus on automation and smart technologies is paramount, it is essential to consider the human factors and user experience in the design process, as these elements significantly influence the effectiveness of smart systems [5].

1.2. Overview and background of Internet of Things

The ability of individual devices to interact with one another is a prerequisite for the Internet of Things (IoT) to become a reality [35]. This is why communication is the most essential element in its development. Unless one of these abilities is expressly needed by a gadget, all the other capabilities, such as detecting, manoeuvring, collecting, storing, and processing information, are completely useless [36].

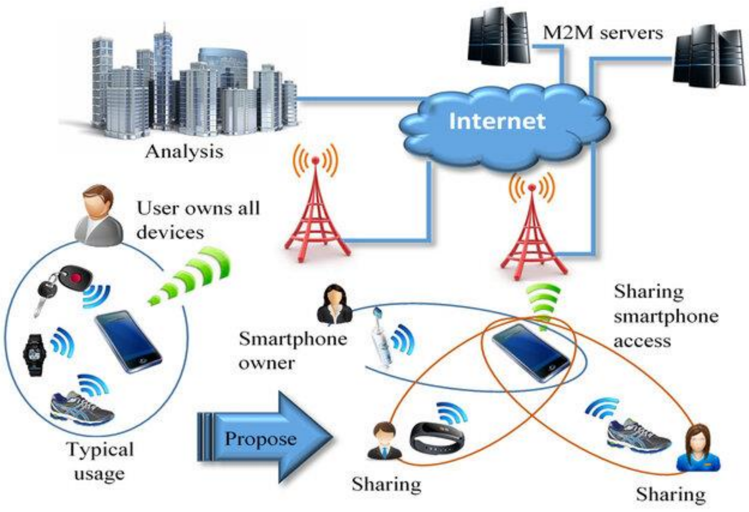

Figure 1: IoT Communication Schema [21]

However, to qualify as an Internet of Things device, a piece of hardware must be able to communicate with other devices [6]. Because physical and link layer communication in IoT may be performed in many different ways, the way this communication is conducted is not as important as it formerly was. Diagram 1 illustrates the fundamental framework behind how the Internet of Things operates and how devices connect [37]. Figure 1 depicts a typical use scenario as well as a potential solution for sharing an access point so that several users in a service region may connect to the Internet. In recent years, the primary trend of the Internet of Things (IoT) has been the accelerated creation of a large number of devices that can be linked to and controlled by the Internet [38]. Because there are so many different uses for IoT technology, the specifications of each device might vary widely from one another, while most of them have the same fundamental capabilities [39].

First and foremost, interconnectivity is the defining trait of the Internet of Things (IoT), given that the foundation of the whole concept and notion behind it is the idea that everything should be able to be connected to everything else (despite the traffic going through different networks) [40]. What a machine or device can do, when it can do it, and how often it can do it are all limited by the efficiency of its central processing unit (CPU), memory, and power. Things-related services tackle issues revolving around devices that have these limitations [41].

- Things-related Services - It provides things-related services within the constraints of things, such as privacy and data interchange consistency between virtual and physical things [42].
- Heterogeneity - IoT devices have different hardware and use different networks, but even through different networks, they can still communicate with other devices [43].

Because there are many distinct protocols now in use, supporting heterogeneity is going to be one of the most difficult tasks. Interacting with a large number of devices across a variety of networks will be challenging from both a technical and a security point of view since protocols might change depending on which interface the device uses to connect (e.g., wide cellular radio, Ethernet, or Wi-Fi) (Figure 2).

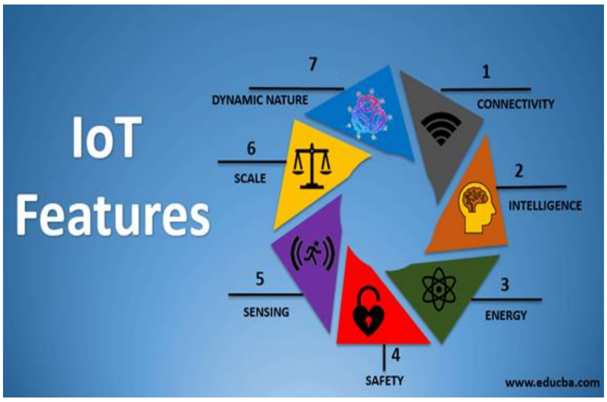

Figure 2: Characteristics of IoT [22]

Therefore, several standards are pertinent to the Internet of Things, such as privacy and data protection. If everything is connected, then there will be numerous security risks, which will make confidentiality, integrity, accessibility, and authenticity more significant–especially as more data and services will be available and as an increasing number of activities will depend on this information. If everything is connected, then everything will be connected [44]. If everything is connected, then everything will be connected. If everything is connected, then everything will be connected. Privacy is an essential component of security because the data gathered by a sensor, for instance, can include information that is very sensitive or personally identifiable [45]. Integrity has to be considered at every stage (sensing, storage, transmission, etc.), which means that Internet of Things security needs to be flexible enough to accommodate a wide variety of devices and networks [46].

- Dynamic changes - Device status may alter dynamically, so the number of devices may differ. (Device states: connected, disconnected, waking up, and sleeping) [47]
- Enormous scale - The number of devices working and interacting will be greater than the number of devices on the existing Internet. Most of this communication will be device-to-device rather than device-to-human [48].

1.3. Principles and theory of automation

Power is essential for each work that is to be conducted by an automated system, as such activities are expected to be useful. Even though there is a great deal of diverse types of power that may be accessed, the majority of the automated systems that are in use today make use of electricity as their source of power [49]. Electrical power is the most versatile form of power because it can be readily generated from other sources (such as fossil fuel, hydroelectric, solar, and nuclear power). It can also be readily converted into other forms of power (such as mechanical, hydraulic, and pneumatic power) in order to carry out useful work [50]. In addition, it is possible to store electrical energy in batteries that both have an elevated level of performance and a long lifespan (Figure 3).

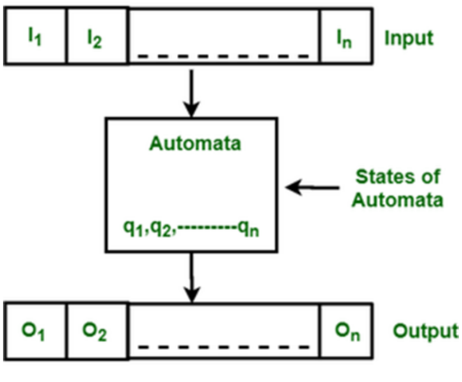

Figure 3: Automation process [23]

Processing and placement are the two basic types of operations that automated systems typically conduct. Other activities that automated systems may do include transfer [51]. The first category is known as processing. (2) Transfer and placement are the second kind. In the first possible scenario, some entity undergoes some processing action because of energy being provided to it in order to do the action [52]. The process may include the bending of metal, the moulding of plastic, the switching of electrical signals within a communication system, or the processing of data inside an information system that is managed by a computer [53]. All these operations require the expenditure of energy in order to move the thing (such as the metal, the plastic, the electrical impulses, or the data) from one state or condition into another state or condition that is superior in terms of its utility or value [54]. This movement can be thought of as moving the thing from a state in which it is less useful to a state in which it is more useful [55].

The second kind of activity, which is referred to as transfer and placement, may most readily be seen in automated production systems that are designed to conduct work on a product [56]. In instances like these, the item being processed must be transported (transferred) from one area to another during the sequence of phases involved in the manufacturing process. When the product is being processed in several stages, it is often required to position the product exactly at each stage [57]. In the context of automated communications and information systems, the terms "transfer" and "positioning" refer, respectively, to the movement of data (or electrical signals) between the various processing units and the delivery of information to output terminals (such as printers, video display units, and so on) for human interpretation and application. In other words, "transfer" refers to the movement of data between the various processing units, while "positioning" refers to the delivery of information to the output terminals [58].

2. Methodology And the Integral Architecture of The Proposed Development Framework for The Project

For this project, we have implemented different technologies and frameworks. For the backend, we have used Spring Boot integrated with the Angular framework frontend. The database is created in PostgreSQL, and CI/CD pipelines are created to test the code automatically.

2.1. Spring Boot – Backend

The Spring Framework includes a component known as Spring Boot. It is used to construct Spring-based applications that are stand-alone and production-grade with as little work as possible. The fundamental Spring Framework serves as the basis for its development [59]. The architecture of Spring Boot is tiered, and each layer interacts with the one right below or above it in the hierarchical structure [60]. Before we can handle the Spring Boot Architecture, we need to have a solid grasp of the many layers and classes that are included in it [61]. The following describes each of the four layers that make up Spring Boot in Figures 4 and 5:

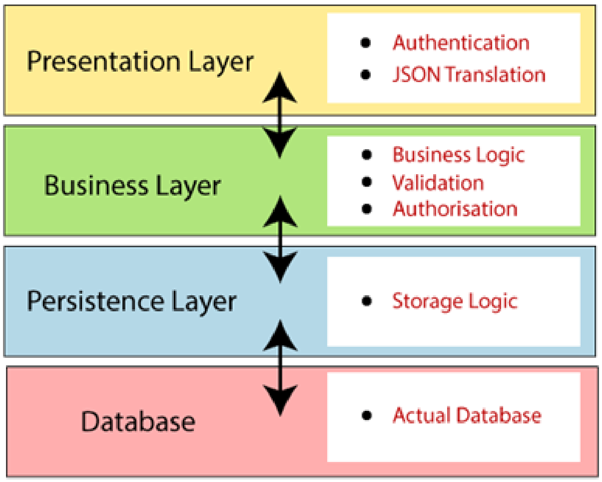

Figure 4: Architecture of spring boot [24]

Presentation Layer: The presentation layer is responsible for handling HTTP requests, converting JSON parameters to objects, authenticating requests, and then passing them on to the business layer. In a nutshell, it is made up of views, which are considered to be part of the front end.

Business Layer: The business layer is responsible for managing all of the business logic. It takes advantage of the services offered by the underlying data access layers and is made up of service classes. Additionally, authorization and validation are carried out via it.

Layer of Persistence: The persistence layer is responsible for translating business objects from and to database rows. It also stores all the logic for storing data.

Database Layer: Operations known as create, retrieve, update, and delete (CRUD) are carried out in this layer of the database.

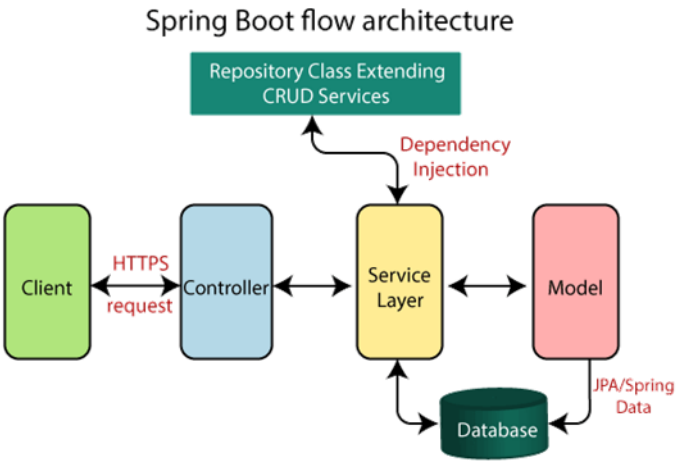

Figure 5: Architecture flow of spring boot [25]

- Currently, we have access to view classes, validator classes, and utility classes.
- Spring Boot takes advantage of all of Spring's modules, such as Spring MVC and Spring Data, among others. The design of Spring Boot is identical to the architecture of Spring MVC, with one key difference being that Spring Boot does not need the use of classes denoted as DAO or DAOImpl.
- Produces a data access layer and carries out CRUD operations on the data.
- The client sends out HTTP requests (PUT or GET).
- The request is sent to the controller, and the controller is responsible for mapping and handling the request once it is received. After that, it initiates a call to the service logic if necessary.
- All the business logic operations are carried out at the service layer. It applies the logic to the data that is mapped to JPA using model classes and then retrieves the results.
- The user will be given a JSP page to view if there are no errors throughout the process.

2.2. Angular – Frontend

A client application may be built with Angular using HTML and either JavaScript or another language, such as TypeScript, which compiles with JavaScript. The framework is formed of a number of libraries, some of which are required while others are elective. Composing HTML templates, building component classes to manage those templates, adding application logic in services, and enclosing components and services in modules are the steps involved in the development of an Angular application (Figure 6).

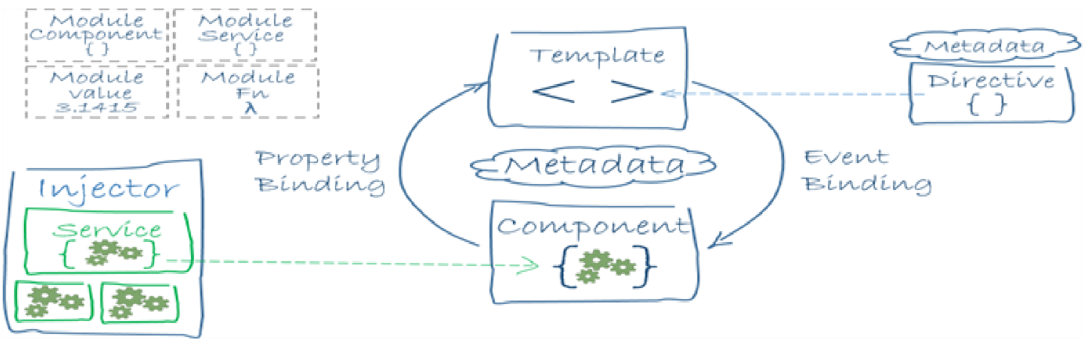

Figure 6: Architecture of Angular

At this point, the application starts by bootstrapping the root module. Such application's content will be shown in a browser by Angular, and it will react to user interactions in accordance with the instructions that the development team has supplied. For now, focus on the big picture.

2.3. Infrastructure – Docker and Kubernetes

When the Enable Kubernetes box is checked, the installation of a Kubernetes cluster with a single node is initiated when the Apply & Restart button is pressed. This is all that a developer requires (Figure 7).

What precisely is going on behind the scenes?

On the backend of the Docker Desktop and inside the virtual machine, the following events are triggered:

- Download and installation of Kubernetes internal components.
- Initialization of the cluster.
- Installation of extra controllers for networking and storage.

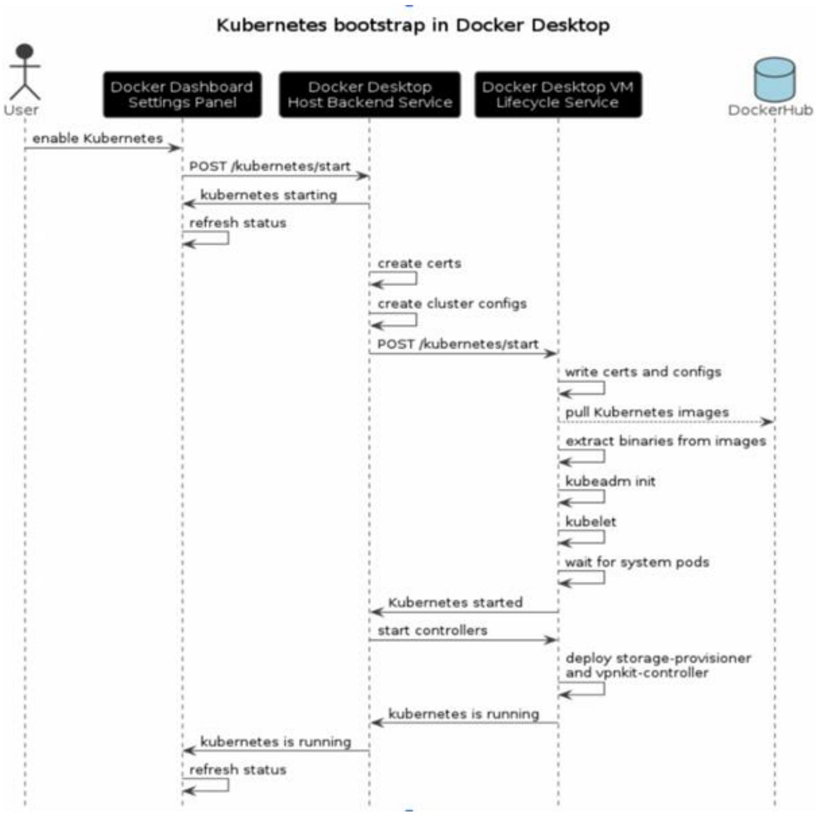

Figure 7: Architecture of Kubernetes

2.3.1. Generating cluster certs, keys and config files

When Kubernetes must establish authorized connections between its internal components and the outside world, it needs to use certificates and keys. Docker Desktop will take care of creating the necessary server and client certificates for the most important internal services, which include kubelet (the node manager), service account management, front proxy, API server, and etch components. Because Docker Desktop uses kubeadm to install Kubernetes, Docker Desktop must establish the kubeadm runtime and configure the cluster-wide settings. This configuration contains the topology of the cluster's network, certificates, and the endpoint of the control plane, among other things. It employs a naming scheme that is unique to Docker Desktop and is not modifiable by the user. At the same time, the global endpoint of the cluster is utilizing the DNS name. https://kubernetes.docker.internal:6443, the current context, user, and cluster names are always set to docker-desktop. The Kubernetes control plane is linked to the default port, which is 6443. Communication with the control plane is made easier since Docker Desktop forwards this port on the host, just as it would be if the control plane were installed directly on the host.

2.4. CI/CD Jenkins

- The phrases “continuous delivery and development” (CI/CD) and “continuous integration” (CI) are synonymous and may be used interchangeably.
- It is an element of the workflow for DevOps, and it is responsible for the generation of application delivery pipelines that are more effective thanks to continuous testing, integration, deployment, and updating of code.
- Continuous Integration and Continuous Delivery are essential aspects of cloud computing, and Microservices and Containerization make it feasible to perform these processes more efficiently.
- Continuous Integration and Continuous Delivery as a Service give service providers the ability to develop automated pipelines that companies can then construct on top of.

3. The Detailed Implementation Framework

For the implementation of this study project, we have used different technologies to develop it. Using Spring boot and Angular, the backend and front end run in perfect condition. In the following subsection, we present first the architecture of the implementation framework using UML analysis and diagrams. Here’s a comprehensive structure for the system under design.

3.1. UML-Based Design of The Proposed Integral System

The Security module is essential for maintaining the integrity and protection of systems. It consists of the Authentication, Authorization, and EncryptionService classes. The authentication class interacts with the authorization class to validate user permissions and ensure secure access to resources. The Encryption Service enhances data security by enabling robust encryption mechanisms. In the Docker ecosystem, the Container class relies on the Image class to provide the necessary environment for application deployment. The Volume class is closely associated with the Container to handle persistent storage, while the Network class facilitates communication between containers, ensuring seamless connectivity within the Docker environment. The Server module includes the HTTPServer and WebSocketServer classes, which are responsible for handling different types of client requests. Both server classes interact with the Controller class to manage and respond to incoming requests, ensuring efficient and scalable server operations.

Jenkins is a vital component in DevOps processes, consisting of the Pipeline, Job, and Build classes. The Job class triggers a Build within the Pipeline, facilitating automated software development, testing, and deployment workflows. The IoT domain includes the Sensor, Actuator, and Gateway classes. The Sensor class sends data to the Gateway, while the Actuator class receives commands from the Gateway to perform specific actions. This interaction enables seamless communication and automation within IoT ecosystems. In Angular applications, the Component, Service, and Module classes are pivotal. The Component class utilizes the Service class to perform business logic, while the Module class aggregates multiple Components and Services, creating a modular and maintainable application structure.

Spring Boot operates on the principle of layered architecture with Controller, Service, and Repository classes. The Controller class interacts with the Service class to handle business logic, while the Service class communicates with the Repository class to perform database operations. The Simulator module includes the SimulationEngine, Scenario, and Result classes. The Scenario class runs within the SimulationEngine, which processes the simulation and outputs the Result. This module is integral for modelling and analyzing complex systems. The PostgreSQL database system incorporates the Database, Table, and Query classes. The Table resides within the database, and the Query class operates on the Table to retrieve or manipulate data, ensuring efficient data management. The Controller module comprises the Command and Response classes. The Command class generates a Response based on the instructions provided, facilitating streamlined communication between system components.

In Pipelines, the Task, Stage, and Step classes define the workflow. Each task consists of multiple Stages, and each stage includes several Steps, creating a structured and efficient process for executing operations. The DevOps module integrates CI/CD, Monitoring, and Logging classes. The CI/CD class collaborates with Jenkins to automate build and deployment pipelines while also utilizing Monitoring and Logging for performance tracking and issue resolution. For Smart Vehicles, the VehicleController, NavigationSystem, Sensor, and Actuator classes work together to manage vehicle operations. The VehicleController communicates with the NavigationSystem for route guidance and interacts with Sensors and Actuators for data collection and action execution. Finally, Kubernetes comprises the Cluster, Node, Pod, and Service classes. The cluster manages multiple Nodes, which host several Pods. The Pods are exposed through a Service, enabling efficient container orchestration and scalability. This comprehensive integration of components and relationships across domains like security, IoT, DevOps, and smart systems demonstrates the interconnected nature of modern technology ecosystems. Here’s a simplified representation of how these components might be related (Figure 8):

```
+------------------+     +------------------+
|   Controller   |<---->|     Server      |
+------------------+     +------------------+
          ^                  ^
          |                  |
+----------+---------+   +-------+-------+
   |    Jenkins       |  |    IoT      |
 +--------------------+   +---------------+
          ^                  ^
          |                  |
+----------+---------+   +-------+-------+
  |    Security      |  |   Docker    |
 +--------------------+   +---------------+
          ^                  ^
          |                  |
+----------+---------+   +-------+-------+
|  Spring Boot     |  |  PostgreSQL |
 +--------------------+   +---------------+
          ^                  ^
          |                  |
+----------+---------+   +-------+-------+
|  Angular        |  |   Simulator     |
 +--------------------+   +---------------+
          ^                  ^
          |                  |
+----------+---------+   +-------+-------+
  |  Pipelines      |  |  DevOps     |
 +--------------------+   +---------------+
          ^                  ^
          |                  |
+----------+---------+   +-------+-------+
|  Smart Vehicles  |  | Kubernetes  |
 +--------------------+   +---------------+
```

Figure 8: UML activity diagram as a flow chart in steps

Activity Diagram: Automation Facilities and Services Integration

Start

Security Authentication
- Action: User logs in.
- Action: System verifies credentials.

Initialize Docker Containers
- Action: Docker containers are started.
- Action: Necessary images are pulled and built.

Server Setup
- Action: Start HTTP Server.
- Action: Start WebSocket Server.

Jenkins Pipeline Execution
- Action: Trigger Jenkins job.
- Action: Execute build and deploy steps.

IoT Device Initialization
- Action: Sensors and Actuators are initialized.
- Action: Gateway establishes communication.

Angular Frontend Setup
- Action: Load Angular modules.

- Action: Initialize components and services.

Spring Boot Backend Setup
- Action: Initialize Controllers.
- Action: Start Services and Repositories.

Simulation Engine Execution
- Action: Load simulation scenarios.
- Action: Run the simulation engine.

Database Connection
- Action: Connect to the PostgreSQL database.
- Action: Load data tables.

Controller Command Execution
- Action: Execute command.
- Action: Generate a response.

Pipeline Execution (CI/CD)
- Action: Execute CI/CD pipeline.
- Action: Monitor pipeline execution.

DevOps Monitoring and Logging
- Action: Monitor system performance.
- Action: Log system events.

Smart Vehicle Control
- Action: Vehicle Controller sends commands.
- Action: Navigation System processes inputs.
- Action: Sensors and Actuators respond to commands.

Kubernetes Cluster Management
- Action: Manage Kubernetes cluster.
- Action: Scale pods and services.

End

4. Discussion of the Design and Results

Integration of Security: The integration of security components, including authentication, authorization, and encryption services, ensured robust protection for all interactions within the system. By validating user credentials and permissions, the security mechanisms prevented unauthorized access, safeguarding sensitive data and critical operations. Encryption services further enhanced data integrity and confidentiality, which is crucial in environments like ships and smart vehicles where data breaches can have severe consequences.

Docker Containerization: Docker's containerization allowed for efficient deployment and management of services. Containers encapsulated the various components, such as the server, Jenkins, and Spring Boot applications, providing isolated and consistent environments. This modular approach facilitated smooth updates and scaling of individual services without impacting others, thus enhancing the system's overall reliability and maintainability.

Server Operations: The deployment of HTTP and WebSocket servers enabled seamless communication between different components. The HTTP server handled RESTful API requests, while the WebSocket server supported real-time data transfer, which is essential for responsive interactions in smart vehicles and automation systems on ships. This dual-server setup ensured optimal performance and user experience.

Jenkins Pipeline Automation: Jenkins played a pivotal role in automating the CI/CD pipeline. By integrating Jenkins, the system could automatically build, test, and deploy updates, significantly reducing manual intervention and errors. This automation streamlined the development process, ensuring that new features and fixes were rapidly and reliably deployed, keeping the system up-to-date and functional.

IoT Integration: The incorporation of IoT devices, such as sensors and actuators, provided real-time monitoring and control capabilities. These devices communicated through gateways, enabling the central system to gather data and execute commands efficiently. The real-time nature of IoT integration is particularly valuable in dynamic environments like ships and smart vehicles, where conditions can change rapidly.

Angular Frontend Development: Using Angular for the front end ensured a dynamic and responsive user interface. The framework's modular structure and component-based architecture facilitated the development of a user-friendly and intuitive interface. This was crucial for enabling operators and users to interact with the system efficiently, whether for monitoring or controlling various functions.

Spring Boot Backend Development: Spring Boot was instrumental in developing the backend services. Its robust framework provided a seamless environment for creating RESTful services and managing business logic. The integration of Spring Boot ensured that the backend was scalable, secure, and easy to maintain, which is essential for complex systems with numerous interconnected services.

Simulation Capabilities: The simulation engine allowed for testing various scenarios and conditions, providing insights into system behaviour and performance. This capability is vital for both ships and smart vehicles, where predicting and preparing for different operational situations can enhance safety and efficiency. The simulation results informed improvements and optimizations, ensuring the system's robustness.

PostgreSQL Database Management: PostgreSQL served as the backbone for data storage and retrieval, handling the vast amounts of data generated and used by the system. Its reliability and performance ensured that data was stored securely and could be accessed efficiently. The use of PostgreSQL supports complex queries and data analysis, which are essential for monitoring and decision-making processes.

Controller Command Execution: The controller facilitated command execution and response generation, enabling the system to interact with various components and execute tasks. This central command functionality ensured coordinated operations and effective control over all integrated services, enhancing overall system performance and reliability.

Pipelines and DevOps: The pipelines managed by Jenkins and other DevOps tools ensured continuous integration and delivery of updates. The monitoring and logging functionalities provided by DevOps tools helped maintain system health and performance, identifying and addressing issues promptly. This continuous improvement cycle kept the system agile and adaptive to new requirements and challenges.

Smart Vehicle Control: The smart vehicle control components, including the Vehicle Controller and Navigation System, ensured precise control and navigation capabilities. The integration with sensors and actuators provided real-time feedback and adjustments, enhancing the vehicle's responsiveness and safety. This comprehensive control system is crucial for the autonomous operation of smart vehicles.

Kubernetes Management: Kubernetes facilitated the orchestration and management of containers, ensuring that the system could scale and adapt to changing demands. The clustering capabilities of Kubernetes provided high availability and fault tolerance, ensuring that the system remained operational even in the face of hardware or software failures.

5. Conclusions

In summary, the Internet of Objects (IoT) is a notion that aims to link the digital world of information technology with the physical world of actual things. Technologies like RFID, sensors, and detectors have made our lives much better and more pleasant. The Internet of Things has the potential to significantly increase data accessibility in every industry around the world, and it will almost certainly cause companies and organizations to undergo fundamental change. As a result, figuring out ways to put the Internet of Things capabilities to use is expected to contribute to the strategic objectives of most technology businesses, regardless of the industries in which they specialize. The concept of the Internet of Things, as well as making it feasible to depict, comprehend and implement the simulation and create the automated IoT devices, were all aspects of the study that addressed the topic and worked toward the aim. To demonstrate how the Internet of Things (IoT) can assist businesses and other organizations in managing a variety of risks, the primary objective was to provide a fundamental illustration of how IoT can be utilized to assist during times of fire emergency or even during a break-in at the time. At the conclusion of this study, every one of the issues that we have posed has been addressed, and an IoT spring boot simulation has been created. The integration of these diverse components into a cohesive system demonstrated the power of modern automation technologies. Each component played a critical role in ensuring the system's functionality, security, and performance. The results highlighted the importance of a well-orchestrated approach to integrating various technologies, ensuring that they work harmoniously to achieve the desired outcomes. This comprehensive integration sets a strong foundation for advanced automation in ships and smart vehicles, paving the way for future innovations and enhancements.

Acknowledgement: The support of my co-author is highly appreciated.

Data Availability Statement: The research contains data related to IoT, Modern Software, and Infrastructure analytics and associated metrics. The data consists of views and dates as parameters.

Funding Statement: No funding has been obtained to help prepare this manuscript and research work.

Conflicts of Interest Statement: No conflicts of interest have been declared by the authors. Citations and references are mentioned in the information used.

Ethics and Consent Statement: The consent was obtained from the organization and individual participants during data collection, and ethical approval and participant consent were received.

References

1. J. A. Muñoz and R. Pérez, "Design of smart ships for the IoT," in Proceedings of the Second International Conference on Internet of Things, Data and Cloud Computing, Cambridge, United Kingdom, 2017.
2. R. Arendt, A. Kopczyński, and P. Spychalski, "Centralized and distributed structures of intelligent systems for aided design of ship automation," in Information Systems Architecture and Technology: Proceedings of 38th International Conference on Information Systems Architecture and Technology – ISAT 2017, Szklarska Poręba, Poland, pp. 310–319, 2018.
3. Y.-L. Tang and N.-N. Shao, "Design and research of integrated information platform for smart ship," in 2017 4th International Conference on Transportation Information and Safety (ICTIS), Alberta, Canada, 2017.
4. M. E. Vaca-Recalde et al., "Connected and intelligent framework for vehicle automation in smart-ports," IEEE Access, vol. 12, no.10, pp. 120347–120361, 2024.
5. G. Carmosino, A. Bionda, and A. Ratti, "The Design of Human Smart Ships. How design-driven approach can foster future development and innovation in the cruise sector," in AHFE International, California, United States of America, 2023.
6. A. Čolaković and M. Hadžialić, "Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues," Comput. Netw., vol. 144, no. 10, pp. 17–39, 2018.
7. I. Ahmad, S. A. A. Shah, and M. A. Al-Khasawneh, "Performance analysis of intrusion detection systems for smartphone security enhancements," in 2021 2nd International Conference on Smart Computing and Electronic Enterprise (ICSCEE), Cameron Highlands, Malaysia, pp. 19-25, 2021.
8. M. Mahmoud and M. A. Al-Khasawneh, "Greedy Intersection-Mode Routing Strategy Protocol for Vehicular Networks," Complexity, vol. 2020, no.12, p. 4870162, 2020.
9. A. Alarood, N. Ababneh, M. Al-Khasawneh, M. Rawashdeh, and M. Al-Omari, "IoTSteg: ensuring privacy and authenticity in internet of things networks using weighted pixels classification based image steganography," Cluster Computing, vol. 25, no. 3, pp. 1607-1618, 2022.
10. D. Ph and L. M. Mikkelsen, "Enhancing IoT systems by exploiting opportunistically collected information from communication networks," vol.5, no.3, pp.1-12, 2017.
11. M. Dumitrescu, "Advanced software technologies in the ship energy system industry," in 2019 International Conference on ENERGY and ENVIRONMENT (CIEM), Timisoara, Romania, 2019.
12. M. Daekeun, L. Kwangil, and K. Hagbae, "The Design of a Total Ship Service Framework Based on a Ship Area Network," IEICE Transactions on Information and Systems, vol. 93, no. 10, pp. 2858-2861, 2010.
13. J. Sui, G. Yu, and W. Zhang, "The ship monitoring and control network system design," in MSIE 2011, Harbin, China, 2011.
14. F. Dongying, X. Jinchao, W. San-Xi, and L. Zhang, "Design of Integrated Ship Equipment Monitoring and Controlling System Based on Ethernet," Vol. 3, no.4, p.12, 2015.
15. I. Dagkinis, P. M. Psomas, and A. N. Platis, "Enhanced ships operation using Integrated Automation Systems," Int. Jour. Res. Dev, vol. 9, no. 4.S2, p. 54, 2022.
16. L. Huang and Y. Guo, "A General Design for Automation Vehicle Stevedoring Management in Bulk Port," in LISS 2013, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 667–672, 2015.
17. S. Elgafoss, T. Skramstad, and V. Dalberg, "Maritime automation, and emergency handling - software: Design, system integration and quality," in 2009 IEEE Electric Ship Technologies Symposium, Maryland, United States of America, 2009.
18. A. Katie,"Automated Functions: Their Potential for Impact Upon Maritime Sociotechnical System," Chalmers University of Technology, Gothenburg, Sweden, 2020.
19. G. L. Babineau, R. A. Jones, and B. Horowitz, "A system-aware cyber security method for shipboard control systems with a method described to evaluate cyber security solutions," in 2012 IEEE Conference on Technologies for Homeland Security (HST), Massachusetts, United States of America, 2012.

20. H. Hwang and I. Joe, “An IoT-based cloud data platform with real-time connecting maritime autonomous surface ships,” in Lecture Notes in Networks and Systems, Cham: Springer International Publishing, pp. 208–220, 2024.
21. T. N. Vidanagama, D. Arai, and T. Ogishi, “M2M gateway selection scheme for smart wireless devices,” in 2015 18th International Conference on Intelligence in Next Generation Networks, Paris, France, 2015.
22. Educba.com. [Online]. Available: https://cdn.educba.com/academy/wp-content/uploads/2019/12/iot-features.png. [Accessed: 05-Jan-2024].
23. Geeksforgeeks.org. [Online]. Available: https://media.geeksforgeeks.org/wp-content/uploads/20200902201005/automata-300x240.png. [Accessed: 05-Jan-2024].
24. Javatpoint.com. [Online]. Available: https://static.javatpoint.com/springboot/images/spring-boot-architecture2.png [Accessed: 05-Jan-2024].
25. Javatpoint.com. [Online]. Available: https://static.javatpoint.com/springboot/images/spring-boot-architecture2.png [Accessed: 09-Jan-2024].
26. M. A. Al-Khasawneh, S. M. Shamsuddin, S. Hasan, and A. A. Bakar, “MapReduce a comprehensive review,” in 2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE), Shah Alam, Malaysia, pp. 1-6, 2018.
27. S. Markkandeyan, S. Gupta, G. V. Narayanan, M. J. Reddy, M. A. Al-Khasawneh, M. Ishrat, and A. Kiran, “Deep learning based semantic segmentation approach for automatic detection of brain tumor,” International Journal of Computers Communications & Control, vol. 18, no. 4, p.11, 2023.
28. M. A. Al-Khasawneh, A. Alzahrani, and A. Alarood, “Alzheimer’s Disease Diagnosis Using MRI Images,” in Data Analysis for Neurodegenerative Disorders, Singapore: Springer Nature Singapore, pp. 195-212, 2023.
29. O. Ameerbakhsh, F. M. Ghabban, I. M. Alfadli, A. N. AbuAli, A. Al-Dhaqm, and M. A. Al-Khasawneh, “Digital forensics domain and metamodeling development approaches,” in 2021 2nd International Conference on Smart Computing and Electronic Enterprise (ICSCEE), Shah Alam, Malaysia, pp. 67-71, 2021.
30. M. A. Al-Khasawneh, A. Alzahrani, and A. Alarood, “An artificial intelligence based effective diagnosis of Parkinson disease using EEG signal,” in Data Analysis for Neurodegenerative Disorders, Singapore: Springer Nature Singapore, pp. 239-251, 2023.
31. V. Kumar, S. Kumar, R. AlShboul, G. Aggarwal, O. Kaiwartya, A. M. Khasawneh, et al., “Grouping and sponsoring centric green coverage model for internet of things,” Sensors, vol. 21, no. 12, p. 3948, 2021.
32. A. M. Khasawneh, O. Kaiwartya, J. Lloret, H. Y. Abuaddous, L. Abualigah, M. A. Shinwan, et al., “Green communication for underwater wireless sensor networks: Triangle metric based multi-layered routing protocol,” Sensors, vol. 20, no. 24, p. 7278, 2020.
33. M. A. Al-Khasawneh, W. Abu-Ulbeh, and A. M. Khasawneh, “Satellite images encryption review,” in 2020 International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI), Nanchang, China, pp. 121-125, 2020.
34. S. A. A. Shah, M. A. Al-Khasawneh, and M. I. Uddin, “Review of weapon detection techniques within the scope of street-crimes,” in 2021 2nd International Conference on Smart Computing and Electronic Enterprise (ICSCEE), Shah Alam, Malaysia, pp. 26-37, 2021.
35. I. M. Alfadli, F. M. Ghabban, O. Ameerbakhsh, A. N. AbuAli, A. Al-Dhaqm, and M. A. Al-Khasawneh, “Cipm: Common identification process model for database forensics field,” in 2021 2nd International Conference on Smart Computing and Electronic Enterprise (ICSCEE), Shah Alam, Malaysia, pp. 72-77, 2021.
36. H. A. Sukhni, M. A. Al-Khasawneh, and F. H. Yusoff, “A systematic analysis for botnet detection using genetic algorithm,” in 2021 2nd International Conference on Smart Computing and Electronic Enterprise (ICSCEE), Shah Alam, Malaysia, pp. 63-66, 2021.
37. Y. F. Saputra and M. A. Al-Khasawneh, “Big data analytics: Schizophrenia prediction on Apache spark,” in Advances in Cyber Security: Second International Conference, ACeS 2020, Penang, Malaysia, December 8-9, 2020, Revised Selected Papers 2, Springer Singapore, pp. 508-522, 2021.
38. S. A. A. Shah, M. A. Al-Khasawneh, and M. I. Uddin, “Street-crimes Modelled Arms Recognition Technique (SMART): Using VGG,” in 2021 2nd International Conference on Smart Computing and Electronic Enterprise (ICSCEE), Shah Alam, Malaysia, pp. 38-44, 2021.
39. A. M. Alghamdi, M. A. Al-Khasawneh, A. Alarood, and E. Alsolami, “The Role of Machine Learning in Managing and Organizing Healthcare Records,” Engineering, Technology & Applied Science Research, vol. 14, no. 2, pp. 13695-13701, 2024.
40. M. A. Al-Khasawneh, M. Faheem, A. A. Alarood, S. Habibullah, and E. Alsolami, “Towards Multi-Modal Approach for Identification and Detection of Cyberbullying in Social Networks,” IEEE Access, vol. 12, no.6, pp. 90158-90170, 2024.
41. A. O. Alzahrani, M. A. Al-Khasawneh, A. A. Alarood, and E. Alsolami, “A Forensic Framework for gathering and analyzing Database Systems using Blockchain Technology,” Engineering, Technology & Applied Science Research, vol. 14, no. 3, pp. 14079-14087, 2024.

42. M. A. Al-Khasawneh, M. Faheem, A. A. Alarood, S. Habibullah, and A. Alzahrani, “A secure blockchain framework for healthcare records management systems,” Healthcare Technology Letters, IEEE, vol.11, no.6, pp. 461-470 2024.
43. H. A. Al-Asadi, “A Novel and Enhanced Distributed Clustering Methodology for Large Scale Wireless Sensor Network Fields,” Journal of Computational and Theoretical Nanoscience, vol. 16, no. 2, pp. 633-638, 2019. doi: 10.1166/jctn.2019.7782.
44. N. F. H. Husshini, N. A. M. A. Hambali, M. H. A. Wahid, M. M. Shahimin, N. Ali, M. N. M. Yasin, and H. A. AL-Asadi, “Stability Multi-Wavelength Fiber Laser Employing Semiconductor Optical Amplifier in Nonlinear Optical Loop Mirror with Different Gain Medium,” SPIE, vol. 63, no. 5, pp. 1241, 2019. doi:10.1364/AO.506145.
45. N. F. H. Husshini, N. A. M. A. Hambali, M. H. A. Wahid, M. M. Shahimin, M. N. M. Yasin, N. Ali, and H. A. AL-Asadi, “Multiwavelength Fiber Laser Employing Semiconductor Optical Amplifier in Nonlinear Optical Loop Mirror with Polarization Controller and Polarization Maintaining Fiber,” In CAPE2019, Kerala, 2020. doi: 10.1063/1.5142122.
46. N. F. H. Husshini, N. A. M. A. Hambali, M. H. A. Wahid, M. M. Shahimin, M. N. M. Yasin, N. Ali, and H. A. AL-Asadi, “Characteristics of Multiwavelength Fiber Laser Employing Semiconductor Optical Amplifier in Nonlinear Optical Loop Mirror with Different Length Polarization Maintaining Fiber,” In CAPE2019. Kerala, 2020. doi: 10.1063/1.5142121.
47. H. A. Al-Asadi, and H. A. Ahmed, “A Tri-Classes Method for Studying the Impact of Nodes and Sinks Number on Received Packets Ratio of MANETs Routing Protocols,” 2023 15th International Conference on Developments in eSystems Engineering (DeSE), Baghdad & Anbar, Iraq, pp. 521-526, 2023. doi: 10.1109/DeSE58274.2023.10100163.
48. R. Younes, F. Ghosna, M. Nassr, M. Anbar, and H. A. Al-Asadi, “Predicting BER value in OFDM-FSO systems using Machine Learning techniques,” Optica Pura y Aplicada, vol. 55, no. 4, pp. 1, 2022. doi: 10.7149/OPA.55.4.51114.
49. L. Hasan, M. Nassr, M. Anbar, and H. A. Al-Asadi, “Inverted U-shaped Frequency Reconfigurable Microstrip patch antenna for 5G communication systems, Optica Pura y Aplicada,” vol. 56, no. 3, pp. 1-5, doi: 10.7149/OPA.56.3.51153.
50. H. O. M. Al-Jabry, and H. A. Al-Asadi, “Enhancing Wireless Multimedia Sensor Networks with Optimization Algorithms: A Review,” IEEE Al-Sadiq International Conference on Communication and Information Technology, Al-Muthana, Iraq, pp. 153-158, 2023. doi: 10.1109/AICCIT57614.2023.10217875.
51. H. Al-Jabry, and H. A. Al-Asadi, “Enhancing Packet Reliability in Wireless Multimedia Sensor Networks using a Proposed Distributed Dynamic Cooperative Protocol (DDCP) Routing Algorithm,” Iraqi Journal for Electrical and Electronic Engineering, vol.19, no. 2, pp. 158-168, 2023, doi: 10.37917/ijeee.00.0.0.
52. H. H. K. Al-Maliki and H. A. A. Al-Asadi, “Enhancing Performance in Vehicular Ad Hoc Networks: The Optimization Algorithm Perspective,” Proceedings - International Conference on Developments in eSystems Engineering (DeSE), Istanbul, Turkiye, pp. 456–461, 2023. doi: 10.1109/DESE60595.2023.10469611.
53. H. A. Ahmed, and H. A. A. Al-Asadi, “An Optimized Link State Routing Protocol with a Blockchain Framework for Efficient Video-Packet Transmission and Security over Mobile Ad-Hoc Networks,” Journal of Sensor and Actuator Networks, vol. 13, no. 2, p.22, 2023. doi: 10.3390/jsan13020022
54. R. Oak, M. Du, D. Yan, H. Takawale, and I. Amit, “Malware detection on highly imbalanced data through sequence modeling,” in Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security - AISec’19, New York, United States, 2019.
55. B. Senapati and B. S. Rawal, “Adopting a deep learning split-protocol based predictive maintenance management system for industrial manufacturing operations,” in Lecture Notes in Computer Science, Singapore: Springer Nature Singapore, pp. 22–39, 2023.
56. B. Senapati and B. S. Rawal, “Quantum communication with RLP quantum resistant cryptography in industrial manufacturing,” Cyber Security and Applications, vol. 1, no. 12, p. 100019, 2023.
57. B. Senapati et al., “Wrist crack classification using deep learning and X-ray imaging,” in Proceedings of the Second International Conference on Advances in Computing Research (ACR’24), Cham: Springer Nature Switzerland, pp. 60–69, 2024.
58. A. B. Naeem et al., “Heart disease detection using feature extraction and artificial neural networks: A sensor-based approach,” IEEE Access, vol. 12, no.3, pp. 37349–37362, 2024.
59. R. Tsarev et al., “Automatic generation of an algebraic expression for a Boolean function in the basis ∧, ∨, ¬,” in Data Analytics in System Engineering, Cham: Springer International Publishing, Switzerland, pp. 128–136, 2024.
60. R. Tsarev, B. Senapati, S. H. Alshahrani, A. Mirzagitova, S. Irgasheva, and J. Ascencio, “Evaluating the effectiveness of flipped classrooms using linear regression,” in Data Analytics in System Engineering, Cham: Springer International Publishing, Switzerland, pp. 418–427, 2024.
61. Wali, G., and C. Bulla, “Anomaly Detection in Fog Computing: State-of-the-Art Techniques, applications, Challenges, and Future Directions,” Library Progress International, vol. 44, no. 3, pp. 13967–13993, 2024.