

1. For the text below, summarise the problem and suggest a software system architecture solution. The main stakeholders in this case are: - Software system architects Problem, context, and related work Throughout the last couple of years, the research in the field of artificial intelligence (AI) evolved rapidly. That happens thanks to machine learning (ML) and deep learning (DL), which are sub-categories of AI, and concrete applications of AI. In ML, the target is, instead of programming with if-conditions, loops, and instructions as it is done in traditional programming approaches, to let the system learn how to behave. For this purpose, a tremendous amount of data is necessary for training similar to how we, as humans, are learning. Before a newborn is able to recognize a car and distinguish all different kinds of brands and forms, it needs to see hundreds of vehicles repetitively, getting explained by its parents that this is a car. That is why engineers try to apply the way the human brain is working to the software. This approach is referred to as deep learning, where the main idea is to create neural networks with the target to identify specific criteria of a given input in order to recognize it accordingly. The considerable benefit of this approach is its flexibility. Let us consider an example of distinguishing cars according to their brand. The creation of a system with a traditional software engineering approach that is able to identify different brands with its various models, shapes, and colors in any weather and lighting condition is almost impossible. But with AI, this is a solvable task. By feeding the neural network with immense data of all kinds of vehicles in any situation, it is possible to create a system that able to recognize these. Hence, tasks that could not be solved in the past might be solvable with the help of AI. And this is applicable for every primary industry – the consumer market, healthcare, automotive, transportation, retail, logistics, education, agriculture, and more. Some companies are already applying AI to their products and business philosophy to create added values that are enabling them to come up with new business models and new customer target groups. Although there is a significant level of mistrust in AI, particularly from the perspective of losing jobs, existing experiences prove the development of technology does not necessarily lead to higher unemployment. “The countries with the highest robot density – which is Germany, Japan, South Korea, and Singapore - all of them have more than 300 robots per 10,000 workers and have among the lowest unemployment rates. So what we need to learn is, technology and humans combined in the right way will drive prosperity and growth, will drive employment, will drive affordability of products, and will drive demand. And that means, altogether, that if we embrace technology in a responsible way, we can really get there.” - Ulrich Spiesshofer, President, and CEO, ABB Ltd. Nowadays, the majority of companies still do not use AI for their products and services. Although these companies consider AI and are excited about its possibilities (both realistic and unrealistic), they are still failing to adopt AI because of a lack of understanding about the technology itself or due to missing experience with its implementation. Furthermore, the existing architecting approaches so far have ignored or not appropriately dealt with the important question of the adequacy of AI for the existing software systems and for those systems that are to be developed. The entry point of any technology for companies is a system architecture of their software systems. The purpose of the thesis is to compensate for the shortcomings of the state of the art. We aim to address companies and decision-makers with less experience in the field of AI and present an approach that supports them, on an architectural level, in designing an AI powered system while taking the most relevant AI-specific properties under consideration. AI is a disrupting technology with the potential to solve problems that traditional software engineering cannot manage. It is not anymore a simple framework or a plugin. It requires a complete reorganization of the system and introduces too many changes (development, process, system, and software and hardware components). Therefore, it cannot only be adopted by development teams. Still, its effects (benefits and drawbacks, perspectives, and shortcomings) must be discussed on a higher level – hence system

and software architecture. Actual integration and deployment of AI for small- and medium-sized embedded systems companies is still a widely unestablished process. The main reasons for this are: 1. The lack of the understanding of the AI technology itself, which leads to the lack of understanding about 2. potential application areas, and 3. the uncertainty if a migration towards an AI-based solution will be of benefit. The main problem in applying AI in the industry is a lack of a systematic approach for mapping the concerns of stakeholders to potential AI-enabled solutions, which, from the architectural point of view, also discusses consequences in terms of benefits and drawbacks. The domain of AI is very complex and exists in several variations and implementations. It addresses various vertical markets such as medical, consumer, automotive, and the industrial market. Also, its realization can be cloud, fog, or edge-based. Because of the lag of experience of industrial companies that mainly develop embedded hard- and software and due to my personal, professional background, the scope of the thesis will mainly cover AI on the edge and its hardware/software architecture. The problem for companies that are mainly operating in the industrial and embedded domain is a missing basis for decision-making when it comes to the implementation of an AI-approach to their products. For them, the question "Is AI suitable for the existing or new system?" is just barely, if not impossible, to answer confidently. That is where the thesis will provide its contribution. The objective is to support the decision-making process for embedded system companies when adopting AI. This challenge includes getting a profound impression and an increased awareness about technical and business-relevant aspects that need to be under consideration during an AI-implementation. Therefore, engineering techniques are required to get an understanding of the possible impact of implementing AI. Although the SWOT analysis seems to be one of the most common and widely used methods for strategic planning, it has its disadvantages when it comes to addressing particular domain related questions. Its advantage lies in its flexible applicability in different domains and its straightforward usability. But in our case, where we want to evaluate a specific technology, it does not provide any kind of support structure that allows us to incorporate an extension to overcome domain-specific deficiencies. This disadvantage is reflected in the fact that SWOT only describes a vague process of procuring information that is, most of the time, based on subjective impressions (Ashish B. Sasankar and Dr Vinay Chavan. Swot analysis of software development process models.) (Hanieh Kashfi. Adopting cloud computing for enterprise architecture: Swot analysis. International Journal of Advanced Research in Computer Science and Electronics Engineering (IJARCSEE), 6(5), 2017.) In contrast, the Solution Adequacy Check is a research area that can be applied to many different use cases. Fraunhofer RATE, and especially the adequacy check, already provides a solid base when it comes to architecture evaluation. It describes a detailed process of how to acquire data necessary to get a rating for the underlying architecture and thereby enhance the awareness of possible pros, cons, and shortcomings of the architecture. The downside here is that it is not designed for specific domains and technologies that require a more detailed or domain related check as the implantation of AI would do. Therefore it is the aim of the thesis to enhance the adequacy check in such a way that it addresses AI-domain-related circumstances, and that allows evaluating AI-based architectures.

Requirements for the solution Requirement Artifacts • A01: Support Decision Makers who want to adopt AI • A02: Getting an overview of the state of the art ML implementation in edge computing • A03: Identify AI-specific architectural properties • A04: Identify engineering techniques to evaluate the existing system concerning the applicability of AI • A05: Provide a design decision process based on adequacy check or SWOT analysis that maps requirements to potential solutions • A06: Enable a discussion about benefits, constraints, and risks of adopting AI towards technical and business-relevant concerns Requirements • R01: Support companies with existing traditional embedded systems to evaluate the adoption of AI into their current portfolio • R02: Getting an overview of the state of the art ML implementation in edge computing • R03: Identify

AI-specific architectural properties that influence business, system, and hardware architectures. • R04: Identify engineering techniques to evaluate the existing system concerning the applicability of AI • R05: Provide a design decision process based on the Solution Adequacy Check or SWOT analysis that maps requirements to potential solutions • R06: Enable a discussion (analysis and evaluation) about technical and business relevant concerns, such as added-values, benefits, constraints, and risks of adopting AI.

Problem: Companies are failing to adopt AI due to a lack of understanding and experience with its implementation, and the existing architecting approaches have not dealt with the adequacy of AI for existing and new software systems.

Solution: Develop a software system architecture approach that supports decision-makers in designing AI-powered systems while taking AI-specific properties into consideration. The approach should be based on the Solution Adequacy Check and enable the evaluation of existing traditional embedded systems for the adoption of AI. The solution should also provide an overview of the state-of-the-art ML implementation in edge computing, identify AI-specific architectural properties, identify engineering techniques to evaluate the existing system concerning the applicability of AI, provide a design decision process based on the Solution Adequacy Check or SWOT analysis, and enable a discussion about technical and business-relevant concerns, such as added-values, benefits, constraints, and risks of adopting AI.

2. Suggest architecture of a conceptual solution for this problem. We need a C4 system context diagram (<https://c4model.com/#SystemContextDiagram>). Please use text (arrows, dots, different shapes, and different colours) to visualise the solution. Consider the following: - Identify relevant external systems (if any) and add them to the diagram. - Identify relevant external stakeholders (if any) and add them to the diagram. - Identify high level abstract data types transferred between components and stakeholders. - Summarise data types transmitted between components. Add this information to the connections and interfaces between components.

Here is a possible C4 system context diagram for the proposed solution:



In this diagram, we have two main components: the traditional embedded system and the AI-based system. The traditional embedded system represents the existing software systems of companies that want to evaluate the adoption of AI. The AI-based system is the proposed solution that supports decision-makers in designing AI-powered systems while taking AI-specific properties into consideration.

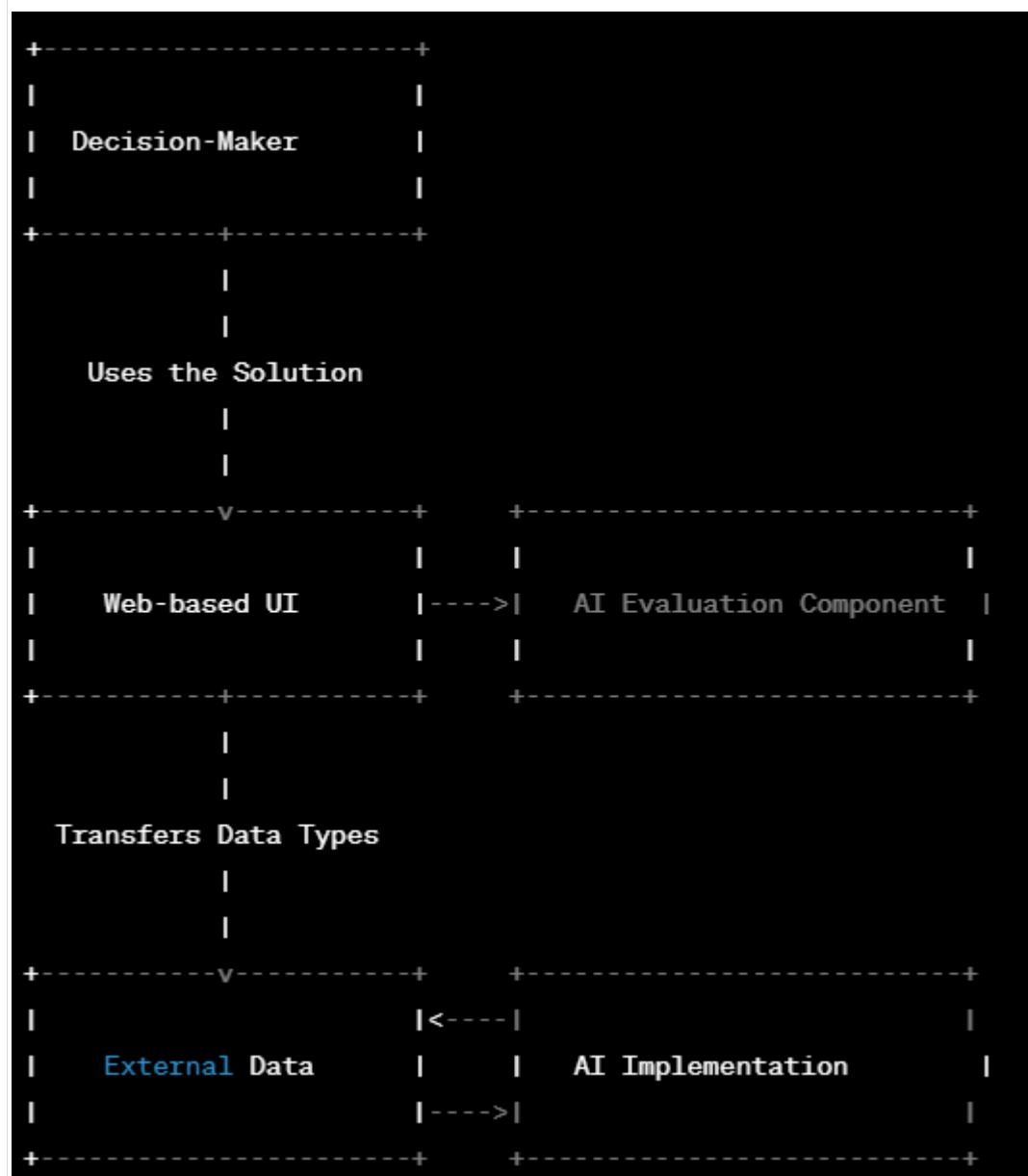
There are two external stakeholders represented as boxes: External Stakeholder A and External Stakeholder B. These stakeholders are relevant because they might have requirements or expectations regarding the use of AI in the systems. For example, External Stakeholder A might be a regulatory agency that requires certain standards for the use of AI in embedded systems, while External Stakeholder B might be a customer that expects AI-enabled features in the products they buy.

The arrows between the components and stakeholders represent the transfer of high-level abstract data types. For example, the traditional embedded system might transfer data types related to sensor measurements or control signals to the AI-based system, while the AI-based system might transfer data types related to the predictions or decisions made by the AI algorithms back to the traditional embedded system. The exact data types would depend on the specific application and use case.

Overall, this C4 system context diagram provides a high-level overview of the proposed solution and its interactions with external stakeholders and systems. It highlights the importance of considering the transfer of data types between components and stakeholders in the design and evaluation of AI-based systems for embedded systems.

3. Suggest architectural components that we need to solve this problem. We need a C4 container solution diagram (<https://c4model.com/#ContainerDiagram>). Please use text (arrows, dots, different shapes, and different colours) to visualise the solution. Consider the following: - Define how the main stakeholder will use the solution. Add it to the diagram.. - Which quality properties are important? Suggest adequate architectural solutions to address them (architectural style, design patterns, design tactics, components and their relationships). Structure the components according to it. - Define data types transferred between components. - Summarise data types transmitted between components. Add this information to the connections and interfaces between components. - Define interfaces between components. Add information about required and provided interfaces to the components, along with the data types. - Document which anti-patterns we need to avoid when implementing this solution.

Here is a possible C4 container solution diagram for the proposed solution:



In this diagram, we have two main components: a web-based user interface (UI) and an AI evaluation component. The decision-maker stakeholder uses the web-based UI to interact with

the solution and make decisions about the adoption of AI in their embedded systems. The AI evaluation component is responsible for evaluating the existing traditional embedded systems and determining their suitability for AI adoption.

Quality properties that are important for this solution include accuracy, scalability, maintainability, and usability. To address these quality properties, we can use the following architectural solutions:

- Architectural style: microservices, to enable scalability and maintainability by separating concerns and enabling independent deployment and scaling of components.
- Design patterns: adapter pattern, to enable the AI evaluation component to work with different types of traditional embedded systems and data formats.
- Design tactics: fault tolerance, to ensure that the solution continues to function even in the presence of errors or failures.
- Components and their relationships: the web-based UI component communicates with the AI evaluation component via a RESTful API, which enables loose coupling and scalability. The AI evaluation component uses machine learning algorithms to evaluate the existing traditional embedded systems, and communicates with external data sources to gather relevant data for the evaluation.

The data types transferred between components include sensor data, control signals, and evaluation results. The web-based UI component sends requests to the AI evaluation component via the RESTful API, and receives evaluation results in JSON format. The AI evaluation component communicates with external data sources to gather relevant data for the evaluation, and may use machine learning algorithms to process the data and generate evaluation results.

The interfaces between components include a RESTful API provided by the AI evaluation component, and required by the web-based UI component. The data types transmitted through the interfaces include sensor data, control signals, and evaluation results, which are communicated in JSON format.

Anti-patterns to avoid when implementing this solution include:

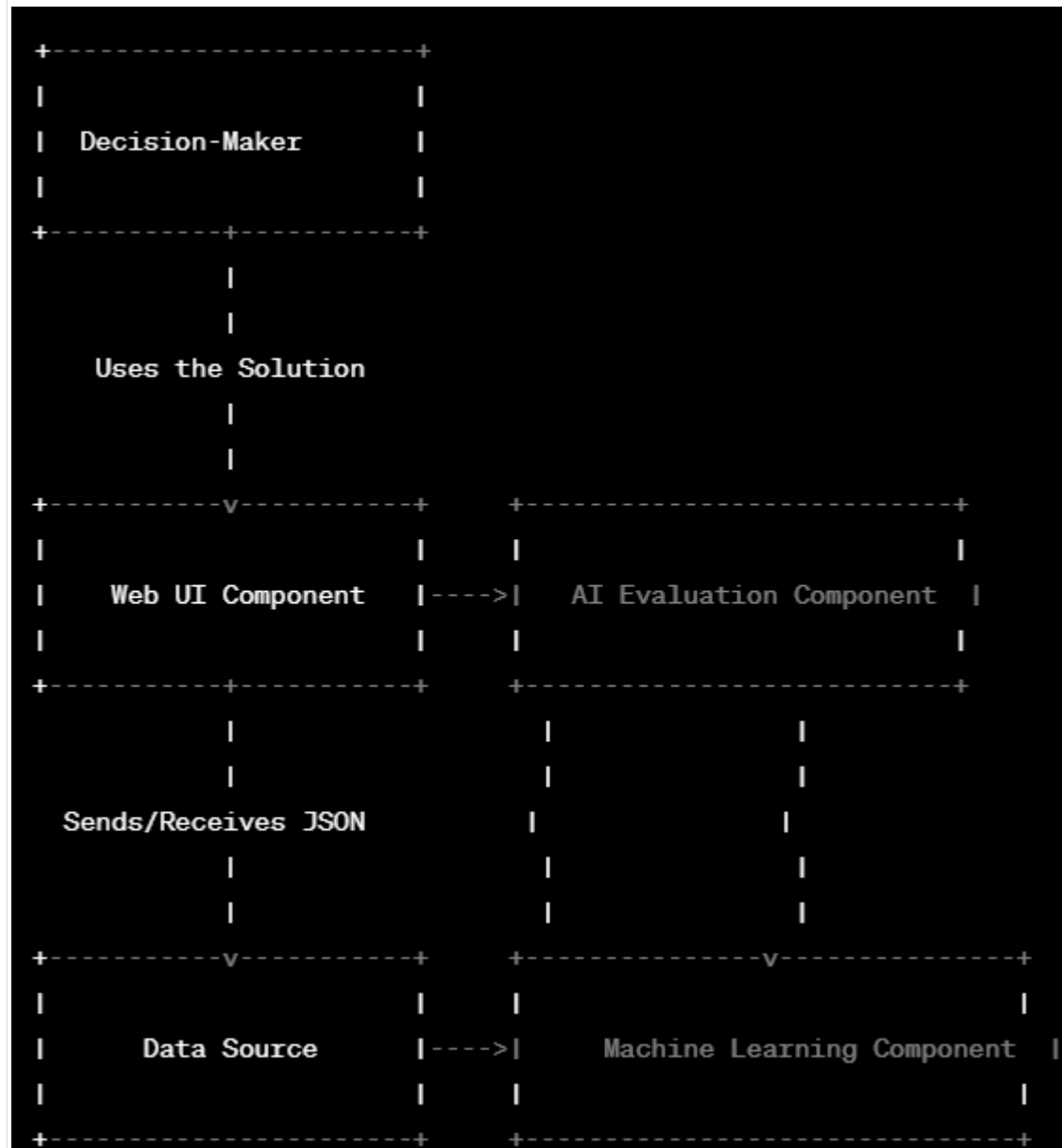
- God object: to avoid having a single component that knows too much about the system and becomes difficult to maintain or scale, we can use microservices to separate concerns and enable independent deployment and scaling of components.
- Tight coupling: to avoid dependencies between components that make the system difficult to maintain or change, we can use loose coupling and standard interfaces, such as RESTful APIs.

4. Suggest architectural components that we need to solve this problem. We need a C4 component solution diagram (<https://c4model.com/#ComponentDiagram>). Please use text (arrows, dots, different shapes, and different colours) to visualise the solution. Consider following:

- Define how the main stakeholder will use the solution. Add it to the diagram.
- Which quality properties are important? Suggest adequate architectural solutions to address them (architectural style, design patterns, design tactics, components and their relationships). Structure the components according to it.
- Define data types transferred between components.
- Summarise data types transmitted between components. Add this information to the connections and

interfaces between components. - Define interfaces between components. Add information about required and provided interfaces to the components, along with the data types. - Document which anti-patterns we need to avoid when implementing this solution.

Here is a possible C4 component solution diagram for the proposed solution:



In this diagram, we have four main components: a web UI component, an AI evaluation component, a data source component, and a machine learning component. The decision-maker stakeholder uses the web UI component to interact with the solution and make decisions about the adoption of AI in their embedded systems. The AI evaluation component is responsible for evaluating the existing traditional embedded systems and determining their suitability for AI adoption. The data source component provides the necessary data for the evaluation process. The machine learning component provides the algorithms and models necessary for the evaluation process.

Quality properties that are important for this solution include accuracy, scalability, maintainability, and usability. To address these quality properties, we can use the following architectural solutions:

- Architectural style: microservices, to enable scalability and maintainability by separating concerns and enabling independent deployment and scaling of components.
- Design patterns: adapter pattern, to enable the AI evaluation component to work with different types of traditional embedded systems and data formats, and the mediator pattern, to enable loose coupling and communication between components.
- Design tactics: fault tolerance, to ensure that the solution continues to function even in the presence of errors or failures, and monitoring and logging, to enable easy diagnosis and resolution of issues.
- Components and their relationships: the web UI component communicates with the AI evaluation component via a RESTful API, which enables loose coupling and scalability. The AI evaluation component uses the adapter pattern to work with different types of data sources, and communicates with the machine learning component to perform the evaluation process.

The data types transferred between components include sensor data, control signals, and evaluation results. The web UI component sends requests to the AI evaluation component via the RESTful API, and receives evaluation results in JSON format. The AI evaluation component communicates with the data source component to gather relevant data for the evaluation, and may use machine learning algorithms to process the data and generate evaluation results.

The interfaces between components include a RESTful API provided by the AI evaluation component, and required by the web UI component. The data types transmitted through the interfaces include sensor data, control signals, and evaluation results, which are communicated in JSON format. The data source component provides the necessary data for the evaluation process, and communicates with the AI evaluation component using a standard interface.

Anti-patterns to avoid when implementing this solution include:

- God object: to avoid having a single component that knows too much about the system and becomes difficult to maintain or scale, we can use microservices to separate concerns and enable independent deployment and scaling of components.
- Tight coupling: to avoid dependencies between components that make the system difficult to maintain or change, we can use loose coupling and standard interfaces, such as RESTful APIs, and design patterns such as the adapter and mediator patterns.

5. Please suggest a toolchain to implement the solution. Consider which tools we can reuse. For the components we need to develop, please suggest a programming language. Please use text (arrows and dots) to visualise the solution. Define data types transferred between components. Summarise data types transmitted between components. Add this information to the connections and interfaces between components.

Here is a possible toolchain to implement the proposed solution:



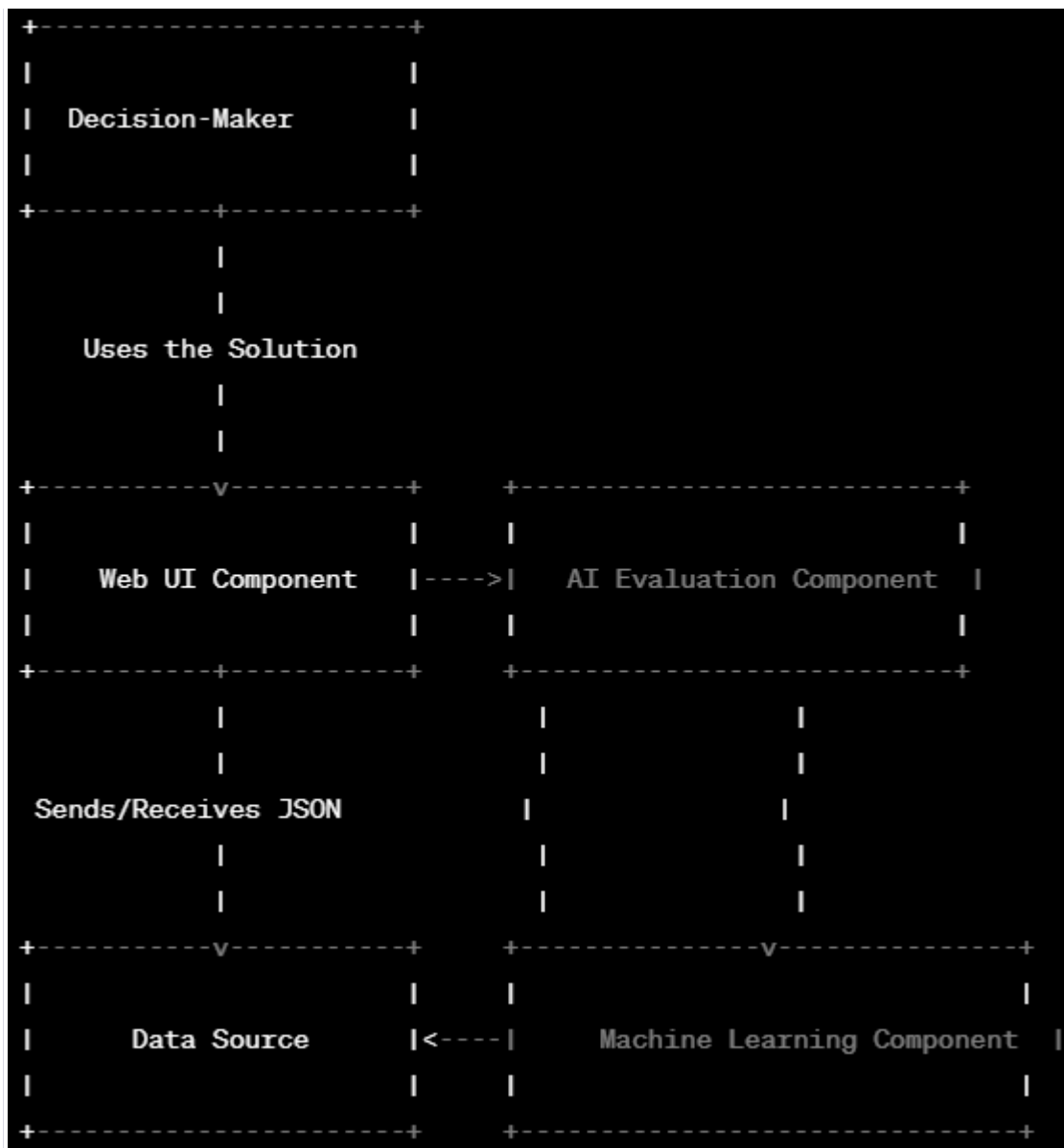
- Web UI component: React.js for building the user interface, and Axios for making HTTP requests to the AI evaluation component.
- AI evaluation component: Node.js for the backend, Express.js for building RESTful APIs, and Python for running machine learning algorithms.
- Data source component: MySQL for storing and retrieving data.
- Machine learning component: Python for running machine learning algorithms.

We can reuse several existing tools and technologies to implement the solution, such as React.js, Node.js, and Python. These tools are widely used and have extensive documentation and community support.

The data types transferred between components include sensor data, control signals, and evaluation results. These data types are communicated in JSON format. Here is a summary of the data types transmitted between components:

- Web UI component to AI evaluation component: requests for AI evaluation, which include metadata about the embedded system and the data to be used for evaluation.
- AI evaluation component to data source component: requests for data to be used for evaluation, which include metadata about the type and format of the data.
- Data source component to AI evaluation component: data to be used for evaluation, which includes sensor data and control signals.
- AI evaluation component to machine learning component: requests to run machine learning algorithms on the data provided.
- Machine learning component to AI evaluation component: evaluation results, which include information about the suitability of the embedded system for AI adoption.

The connections and interfaces between components can be visualized as follows:



In this diagram, the web UI component communicates with the AI evaluation component via a RESTful API, which enables loose coupling and scalability. The AI evaluation component uses the data source component to gather relevant data for the evaluation, and the machine learning component to perform the evaluation process. The connections and interfaces between components are documented with information about the required and provided interfaces, along with the data types transmitted.