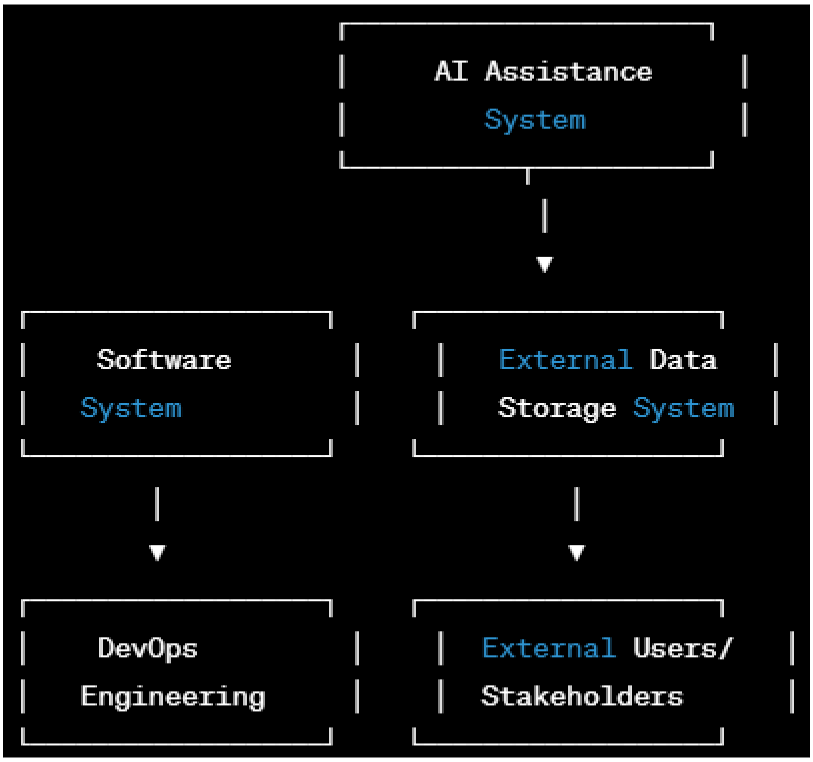
1. For the text below, summarise the problem and suggest a software system architecture solution. The main stakeholders in this case are: - Software system architects - DevOps engineers - Developers Problem, context, and related work Software engineering today is a challenging task. As the complexity of software keeps increasing, software engineers find it ever more challenging to cope with this complexity and produce high-quality source code. Furthermore, developers today spend a significant amount of their time solving similar problems as they struggle with code reuse (e.g. they are aware that the solution exists but it requires tedious changes so re-writing a new code seems easier, and they can misinterpret the requirements, or they are not aware if a solution is available and being used in a different context). Ideally, developers could profit from systems that assist them during the development process by suggesting already existing good (and tested) solutions. This could reduce development efforts. The developers could further benefit from systems that have the capabilities to understand the software requirements. This would enable the creation of assistance mechanisms that can index problem and solution domains and be aware if for an existing or similar problem there exist solutions. We are focusing on supporting developers when dealing with these challenges: 1. Avoidance of repetitive tasks (realizing code that already exists increases time efficiency during development and keeps developers motivated) 2. Avoiding bugs in new code (tested code is often far more valuable because potentially many bugs have been discovered and removed) These challenges are both a drain on motivation that developers have (e.g., influence their productivity and motivation to keep the existing work), and introduce a potential overhead due to all the activities that are related to testing newly implemented solutions. Even if solutions already exist, developers still would need to invest time in understanding them and reusing them. In particular, how well these solutions fit certain requirements. Because understanding source code is a time-consuming task, developers often find it quicker to write the new code. Ideally, there would exist solutions that can understand existing source code (and logic behind it) and relate it to the problems they are solving (requirements). AI is a technology that simulates human intelligence processes through computing systems. In an ideal case, AI is capable to rationalize and take actions towards a specific goal. AI, in this context, is interesting because it can recognize patterns in textual data. In software engineering, both requirements and source code are represented mostly in textual form. Therefore, if we can train AI models to understand requirements and to produce results in a restricted format that matches the semantics of programming languages, this could enable the creation of solutions that assist developers. Ideally, this could lead to the use of AI-enabled technologies as assistance tools. However, this is not an easy task. Gathering requirements for human is challenging as the requirement elicitation process need experience and constant communication between the stakeholders for understanding requirements. Expressing requirements is even hard for software engineers, as they imply many things (e.g. different use cases). They acquire customers' requirements, refine them into tasks and then use AI to assist them with the solutions. Another challenging task is to prioritize goals for development which are often not represented in the requirements and need an expert opinion. When developing software solutions, developers often can choose between various ways of implementation, different coding styles, various algorithms, and different programming languages. For example, if developers want to optimize for a reduced memory footprint, they might use processing algorithms that need more processing power but keep fewer states alive in memory. Finally, it is necessary to test the implementation. This is a challenge, as developers need to be skilled in the implemented technology stack. Gathering requirements can benefit from AI techniques, especially from NLP that can be used to automate this process by extracting valuable information from user requirements by semantic analysis. In an ideal scenario, AI can understand requirements and generate portions of the code to assist developers. These portions can be code snippets or function templates. The focus of this work is

on generating full functions to assist developers. While research in AI and software engineering is potentially providing a variety of solutions in this domain, there still do not exist solutions that can outmatch humans in terms of building program logic and fully understanding the requirements. Even more, although some very impressive solutions can generate source code (e.g., GPT-3), there are no practical solutions that can integrate with development practices and assist developers with writing particularly interesting code and choosing optimal solutions based on the code context and requirements. Therefore, we define our problem statement as follows: Are AI-enabled solutions mature enough to integrate with development processes, index problem and solutions domains, and assist developers by suggesting solutions to common problems? If this is not the case, what is preventing the adoption of AI-based solutions in this way, in software engineering? Therefore, in this thesis, we aim to identify challenges that prevent the adoption of AI from being used as an assistance tool in software development by providing an elaborate list of identified issues and an AI-based approach that can aid software development by generating solutions for already solved problems using user requirements as input. This considers the understanding of how the machine learning algorithms associate source code with natural language and how it understands it. Although there have been several studies focused on source code generation tools that use AI ("Imam, Ayad Tareq and Rousan, Thamer and Aljawarneh, Shadi; An expert code generator using rule-based and frames knowledge representation techniques; 2014", "Soni, Naitik Chetan and Pawar, Dhruv Ashok and Tambe, Namita Sandeep and Deolekar, Rugved Vivek; Automated system for code generation from unstructured algorithm; 2016", "Shah, Meet and Shenoy, Rajat and Shankarmani, Radha; Natural Language to Python Source Code using Transformers; 2021", "Lu, Chao and Cao, Xin and Zhu, Yi and Huang, Tao and Pan, Zhaoming and Li, Xiu; Procedure2Command: an AI-based Nuclear Power Plant Control Command Code Generation Prototype System; 2020", "Liliane do Nascimento Vale and Marcelo de Almeida Maia; Towards a question answering assistant for software development using a transformer-based language model; 2021", "Zhang, Chen and Niu, Xinyi and Yu, Bin; A Method of Automatic Code Generation Based on AADL Model; 2018"), there haven't been significant contributions toward their classification or identification of problems that prevent them to be usable in practice. Requirements for the solution Functional requirements • RQ.1: An approach is able to use dataset for supervised learning and testing & Input • RQ.2: An approach is able to generate a training/testing dataset (For training an algorithm, we would need a specific formatted dataset or a structured dataset. The generated dataset will be the consolidated data collected by a design template (including fields like inputs, function requirements, source code, and output) for users to input, hence making our training data. The same logic can also be applied to gathering the testing data. However, there can be other means to gather datasets, for instance, crawling/scrapping/REST-API calls from open source projects or public/private repositories.) from taking user input via a design template. • The design template will include fields to structure the natural language input provided, such as function input (e.g. parameter integer A, parameter string B, etc.), function requirements (which can include function comments, description, use cases, etc.), function source code and function output (e.g. return type, etc.) • RQ.3: An approach can extract and generate a training/testing dataset from open source projects and public or private repositories (e.g. GitHub, BitBucket, etc.) • RQ.4: An approach is able to persist the consolidated training/testing dataset in a storage system (e.g. database, file storage, etc.) & Pre-Processing • RQ.5: An approach is able to pre-process the stored dataset for training and testing for the AI models (e.g. tokenization, word2vec, stop word removal, etc.) • RQ.6: An approach is able to transform the extracted repository data to a structured format, such that it aligns with the design template dataset • RQ.7: An approach is able to train on Python programming language source code & Training • RQ.8: An approach is able to train on the structured dataset that is prepared using the design template. • RQ.9: An approach is able to train on deep neural networks by using existing models (e.g. GPT2, GPT3, etc.) • RQ.10: An approach is able to understand the requirements in the structured dataset to generate the source code • RQ.11: An approach is able to generate source code for Python programming language based on a set of input requirements (testing design template) & Output • RQ.12: An approach is able to generate multiple source code functions or snippets as a suggestion for the user to select • RQ.13: An approach is able to evaluate the generated source code accuracy based on syntax and logical correctness (by using the following metrics e.g. BLEU score, F-Score, WER, etc.) & Testing and Evaluation • RQ.14: The approach is able to integrate in an IDE (e.g. Eclipse, NetBeans, VS Code, etc.) by a plugin. This will allow the end user to take benefit of the deployed approach in a real-world use case when they are programming in on-going projects. & Deployment • RQ.15: The approach is able to save the trained model in a compressed format (e.g. ZIP, PKL, etc.) in a storage system for testing, evaluating and deploying Quality requirements • Q.1: - Environment: ♣ An approach generating a function source code based on a structured input template ♣ Number of syntax faults = E; Parameters = p - Stimulus: ♣ Re-training the model with hyperparameter tuning ♣ Parameters = q - Response: ♣ Better syntax accuracy ♣ Number of syntax faults = E/2 • Q.2: - Environment: ♣ An approach generating a function source code based on a structured input template ♣ Cyclomatic complexity = CE ♣ Parameters = p - Stimulus: ♣ Re-training the model with hyper-parameter tuning ♣ Parameters = q - Response: ♣ Better syntax accuracy ♣ Number of syntax faults = E/2 • Q.3: - Environment: ♣ An approach generating a function source code in an on-going project ♣ Lack of cohesion in methods (LCOM) = C - Stimulus: ♣ Existing source code in an on-going project which are logically equivalent to the generated source code ♣ Parameters = q ♣ Response Low Cohesion [6] LCOM = 0 • Q.4: - Environment: ♣ An approach generating a function source code in an on-going project ♣ Size of the training dataset and its trained model accuracy; Size = S; Accuracy = A% - Stimulus: ♣ Re-training the model with the increased data size ♣ Size = 2S - Response: ♣ Improved training model accuracy ♣ Accuracy = 2A%

The problem is that software engineering has become increasingly complex, and developers are struggling with code reuse, avoiding repetitive tasks, and avoiding bugs in new code. The solution proposed is an AI-enabled system that can understand requirements and generate code snippets or function templates to assist developers. The system architecture would require functional requirements such as dataset creation, pre-processing, training, and output generation. Quality requirements include hyperparameter tuning, low Cyclomatic complexity, improved accuracy with larger datasets, and integration with IDEs.

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 C4 system context diagram for the conceptual solution:



```
                       ┌─────────────────────┐
                       │   AI Assistance     │
                       │      System         │
                       └─────────────────────┘
                                  │
                                  ▼
┌─────────────────┐    ┌─────────────────┐
│    Software      │    │  External Data  │
│   System         │    │  Storage System │
└─────────────────┘    └─────────────────┘
        │                        │
        ▼                        ▼
┌─────────────────┐    ┌─────────────────┐
│    DevOps        │    │ External Users/  │
│   Engineering    │    │ Stakeholders     │
└─────────────────┘    └─────────────────┘
```

In this diagram, the central component is the AI Assistance System, which provides assistance to developers by generating code snippets or function templates. The system will interact with an external software system and an external data storage system to gather the required data for generating the code. The DevOps Engineering component will be responsible for developing and maintaining the AI Assistance System. External users and stakeholders will interact with the system to provide input, receive output, and evaluate its effectiveness.

The high-level abstract data types transferred between components and stakeholders include natural language input from users and requirements, structured dataset inputs, generated source code outputs, and training/testing datasets.

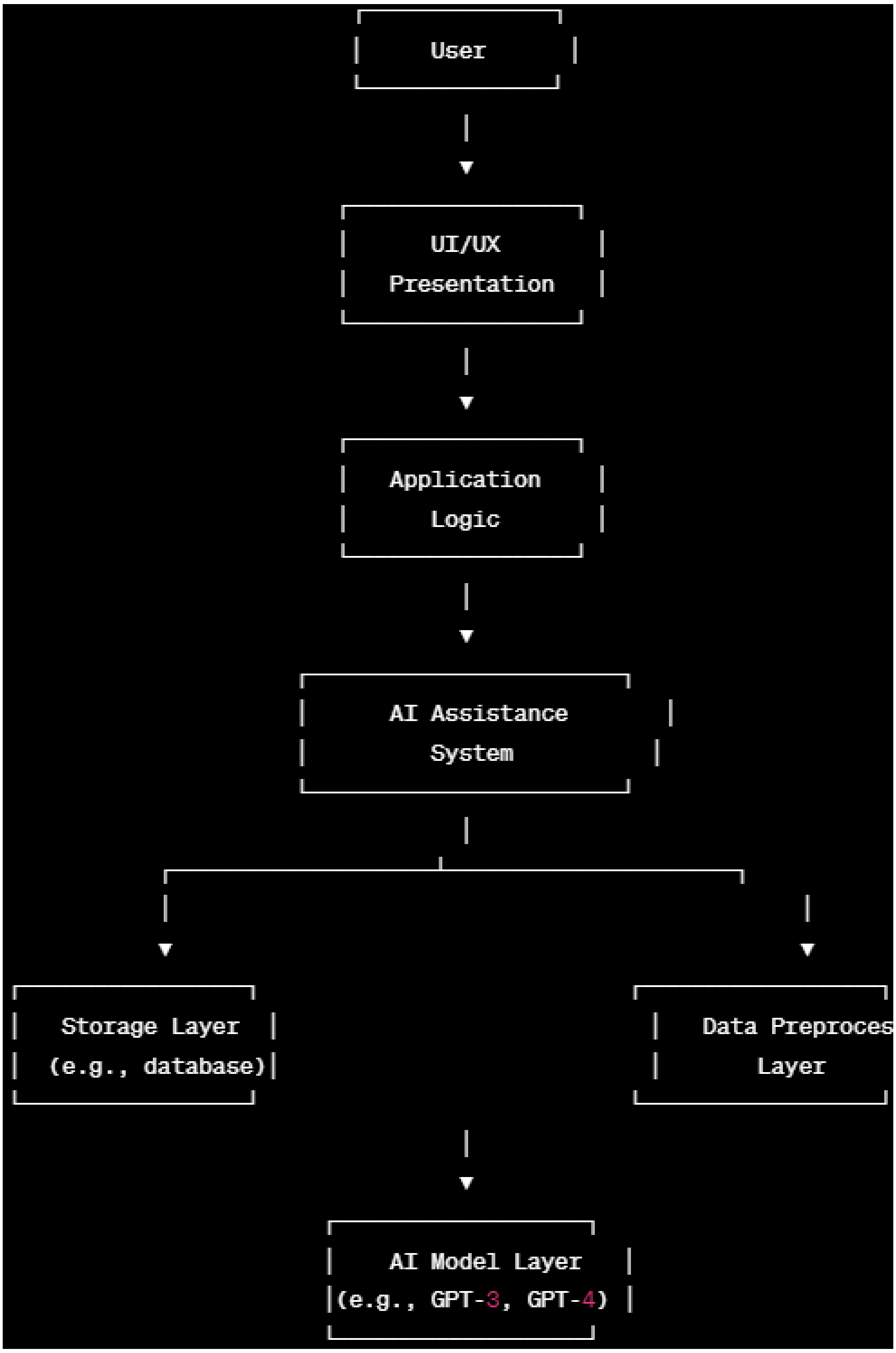
The connections between components and interfaces will involve transferring natural language input and structured dataset inputs to the AI Assistance System for processing. The system will then generate source code outputs and transfer them back to the external software system. The system will also interact with external data storage to gather and store training/testing datasets. External users and stakeholders will provide input to the system through the use of a design template, while output from the system will include generated code snippets or function templates.

Overall, the solution will involve gathering structured datasets, training the AI model, and generating source code outputs that can be used by developers. The solution will require integration with IDEs to facilitate use in real-world projects, and quality requirements will include hyperparameter tuning, low Cyclomatic complexity, and improved accuracy with larger datasets.

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 C4 container diagram for the proposed solution:



```
                 ┌──────────────┐
                 │     User     │
                 └──────────────┘
                        │
                        ▼
                ┌────────────────┐
                │     UI/UX      │
                │  Presentation  │
                └────────────────┘
                        │
                        ▼
                ┌────────────────┐
                │  Application   │
                │     Logic      │
                └────────────────┘
                        │
                        ▼
             ┌────────────────────────┐
             │  AI Assistance         │
             │     System            │
             └────────────────────────┘
                        │
      ┌───────────────────┴──────────────────────┐
      │                                          │
      ▼                                          ▼
┌──────────────────┐                   ┌──────────────────┐
│  Storage Layer   │                   │  Data Preproces
│ (e.g., database)│                   │      Layer
└──────────────────┘                   └──────────────────┘
                        │
                        ▼
               ┌──────────────────┐
               │  AI Model Layer  │
               │(e.g., GPT-3, GPT-4) │
               └──────────────────┘
```

In this diagram, we have identified four main components: User, UI/UX Presentation, Application Logic, and AI Assistance System. External storage and data preprocessing layers are also included.

The user will interact with the system by providing natural language input through the UI/UX Presentation component. The Application Logic component will be responsible for receiving and processing user input and calling the appropriate AI model in the AI Model Layer. The AI Assistance System will use the AI model to generate source code outputs, which will be returned to the Application Logic component for transfer back to the UI/UX Presentation component.

The quality properties that are important for this system include accuracy, efficiency, and usability. To address accuracy, the system will need to use well-trained AI models and have a reliable data preprocessing layer. To address efficiency, the system will need to use efficient algorithms and data structures. To address usability, the system will need to have a user-friendly interface and be easily integrated with IDEs.

The data types transferred between components include natural language input, structured dataset inputs, generated source code outputs, and training/testing datasets.

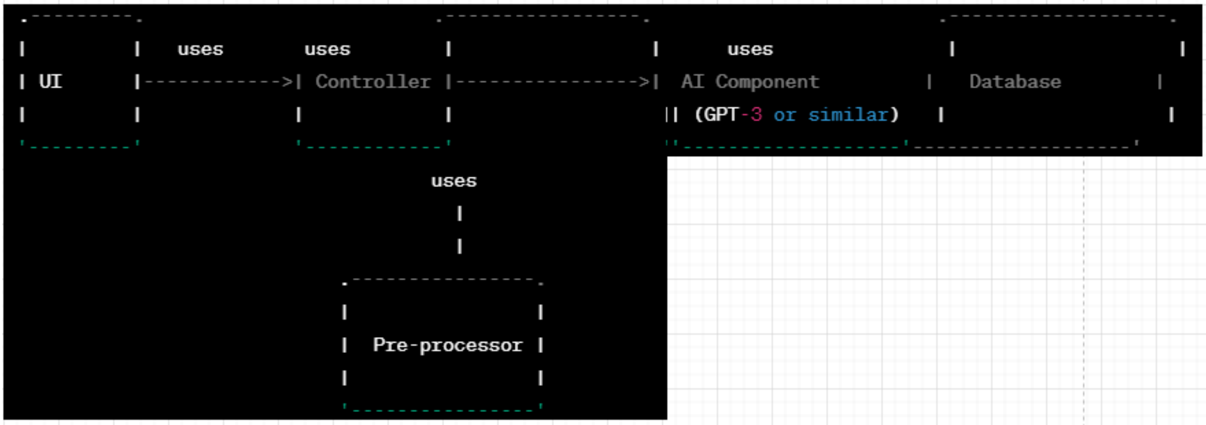
The interfaces between components include the natural language input interface between the User and UI/UX Presentation components, the structured dataset input and generated source code output interfaces between the UI/UX Presentation and Application Logic components, the AI model interface between the Application Logic and AI Assistance System components, and the training/testing dataset interface between the Storage Layer and Data Preprocessing Layer components.

Anti-patterns that need to be avoided when implementing this solution include the God Object anti-pattern, where a single component becomes too large and complex, and the Monolithic architecture anti-pattern, where the system becomes too tightly coupled and difficult to maintain. To avoid these anti-patterns, the system will be designed with a modular architecture, with clear separation of concerns between components and well-defined interfaces. This will allow for easier maintenance, testing, and scalability.

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.

As this is a follow-up question to the previous one, I will continue with the same context and assumptions.

C4 Component Diagram:



```
.---------.                  .-----------------.                    .-------------------.
|         |   uses     uses  |                 |     uses           |                   |
| UI      |------------>| Controller |---------------->|  AI Component       |   Database        |
|         |             |            |                 || (GPT-3 or similar)  |                   |
'---------'             '------------'                 ''------------------'-------------------'
                              uses
                               |
                               |
                        .----------------.
                        |                |
                        |  Pre-processor |
                        |                |
                        '----------------'
```

-
- The UI component will be used by developers to input natural language requirements and receive code suggestions.
- The important quality properties are accuracy and efficiency. To achieve high accuracy, we can use AI-based technologies such as deep learning and natural language processing techniques. To achieve efficiency, we can use a pre-processing component to optimize the input requirements before feeding them into the AI component.
- The Controller component acts as a bridge between the UI and the AI Component, coordinating the interaction between the two components. It also handles the communication with the pre-processor and the database.
- The AI Component will use deep learning models (such as GPT-3) to generate code suggestions based on the input requirements.
- The pre-processor component will optimize the input requirements by performing tasks such as tokenization, stop word removal, and word2vec.
- The database component will store the training and testing datasets used by the AI component for future use.

Data types:

- The UI component will send natural language input data to the Controller component.
- The pre-processor component will receive the natural language input data and output optimized input data to the AI Component.
- The AI Component will receive the optimized input data and output generated code snippets to the Controller component.
- The database component will store training and testing datasets in a structured format.

Interfaces:

- The UI component will provide a user interface for developers to input natural language requirements and receive code suggestions.
- The Controller component will provide interfaces to communicate with the UI component, pre-processor component, and the AI component.
- The pre-processor component will provide an interface to receive natural language input data and output optimized input data.

- The AI Component will provide an interface to receive optimized input data and output generated code snippets.
- The database component will provide an interface to store and retrieve training and testing datasets.

Anti-patterns:

- We need to avoid the anti-pattern of tight coupling between components. Each component should have well-defined interfaces to interact with other components, and they should communicate only through those interfaces.
- We also need to avoid the anti-pattern of over-engineering the system. We should aim to keep the system simple and maintainable by using proven technologies and practices.

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.

To implement the solution, we can use the following toolchain:

- GitHub for version control
- Docker for containerisation
- Jenkins for continuous integration and deployment
- Kubernetes for container orchestration
- Python for development of the AI models and other backend components
- React for the frontend web application

The data types transferred between components include:

- Natural language input/output
- Structured dataset (input design template)
- Training/testing dataset (consolidated data collected by design template)
- Pre-processed dataset for training and testing
- Trained model
- Generated source code snippets or functions
- Syntax and logical correctness metrics

The data types transmitted between components can be summarised as follows:

- User provides natural language input to the frontend web application, which is structured using the input design template and sent to the backend.
- The backend extracts data from open source projects and public/private repositories, consolidates it using the design template, and pre-processes it for training and testing of the AI models.
- The AI models are trained on the pre-processed dataset and generate source code snippets or functions as suggestions for the user.
- The generated source code snippets or functions are evaluated for syntax and logical correctness and provided as output to the frontend web application.

- The frontend web application displays the generated source code snippets or functions to the user.

The interfaces between components can be defined as follows:

- The frontend web application provides natural language input to the backend through a REST API.
- The backend retrieves data from open source projects and public/private repositories through REST API calls.
- The backend stores the consolidated dataset in a database.
- The AI models are trained and evaluated using the pre-processed dataset.
- The backend provides generated source code snippets or functions to the frontend web application through a REST API.

The programming language used for the backend components, including the AI models, is Python. The programming language used for the frontend web application is React.