02_Documentation

ASD-VN-MCAL-CICD-RAC

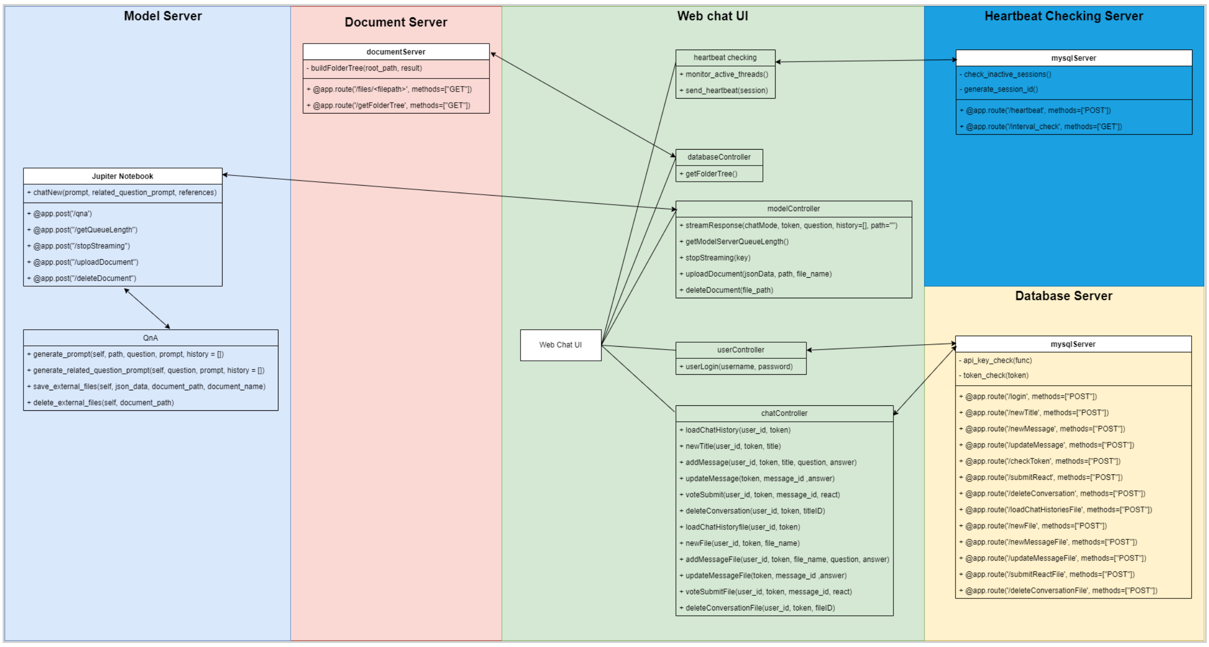
Exported on 02/29/2024

Table of Contents

1 01_System Design

1.1 01_Chatbot System Design

1.1.1 General Class Diagram for ChatBot System


Model Server
Jupiter Notebook
+ chatNew(prompt, related_question_prompt, references)
+ @app.post('/qna')
+ @app.post("/getQueueLength")
+ @app.post("/stopStreaming")
+ @app.post("/uploadDocument")
+ @app.post("/deleteDocument")
QnA
+ generate_prompt(self, path, question, prompt, history = [])
+ generate_related_question_prompt(self, question, prompt, history = [])
+ save_external_files(self, json_data, document_path, document_name)
+ delete_external_files(self, document_path)
Document Server
documentServer
- buildFolderTree(root_path, result)
+ @app.route('/files/<filepath>', methods=["GET"])
+ @app.route('/getFolderTree', methods=["GET"])
Web chat UI
heartbeat checking
+ monitor_active_threads()
+ send_heartbeat(session)
databaseController
+ getFolderTree()
modelController
+ streamResponse(chatMode, token, question, history=[], path="")
+ getModelServerQueueLength()
+ stopStreaming(key)
+ uploadDocument(jsonData, path, file_name)
+ deleteDocument(file_path)
Web Chat UI
userController
+ userLogin(username, password)
chatController
+ loadChatHistory(user_id, token)
+ newTitle(user_id, token, title)
+ addMessage(user_id, token, title, question, answer)
+ updateMessage(token, message_id ,answer)
+ voteSubmit(user_id, token, message_id, react)
+ deleteConversation(user_id, token, titleID)
+ loadChatHistoryfile(user_id, token)
+ newFile(user_id, token, file_name)
+ addMessageFile(user_id, token, file_name, question, answer)
+ updateMessageFile(token, message_id ,answer)
+ voteSubmitFile(user_id, token, message_id, react)
+ deleteConversationFile(user_id, token, fileID)
Heartbeat Checking Server
mysqlServer
- check_inactive_sessions()
- generate_session_id()
+ @app.route('/heartbeat', methods=['POST'])
+ @app.route('/interval_check', methods=['GET'])
Database Server
mysqlServer
- api_key_check(func)
- token_check(token)
+ @app.route('/login', methods=["POST"])
+ @app.route('/newTitle', methods=["POST"])
+ @app.route('/newMessage', methods=["POST"])
+ @app.route('/updateMessage', methods=["POST"])
+ @app.route('/checkToken', methods=["POST"])
+ @app.route('/submitReact', methods=["POST"])
+ @app.route('/deleteConversation', methods=["POST"])
+ @app.route('/loadChatHistoriesFile', methods=["POST"])
+ @app.route('/newFile', methods=["POST"])
+ @app.route('/newMessageFile', methods=["POST"])
+ @app.route('/updateMessageFile', methods=["POST"])
+ @app.route('/submitReactFile', methods=["POST"])
+ @app.route('/deleteConversationFile', methods=["POST"])

1.1.2 System Components

1.1.2.1 Web Chat UI

- The Web Chat UI is the user interface component of your system, built on Gradio and serving as a web server.
- This component provides a seamless and interactive experience for users to engage with the chatbot.
- Users can input queries or messages through the web interface, and the UI is responsible for rendering the responses generated by the underlying language model.
- It acts as a front-end gateway, facilitating user interactions and ensuring a user-friendly experience.

1.1.2.2 Model Server

- The Model Server serves as the core artificial intelligence engine for the chatbot.
- It hosts a Language Model (LLM) responsible for generating responses to user queries. This model may be pre-trained and fine-tuned to understand and generate contextually relevant answers.
- Additionally, the Model Server exposes a QnA service through an API, allowing other services in the system to query the chatbot for information.
- It acts as the brains behind the chatbot's conversational abilities, handling natural language understanding and generation.

1.1.2.3 Database Server

- The Database Server is a MySQL server that plays a crucial role in storing and managing user-related data.
- It stores user IDs and the corresponding conversations, allowing the system to maintain user-specific contexts, preferences, and historical interactions.
- This component facilitates data persistence and retrieval, enabling the chatbot to provide personalized responses and maintain continuity in user conversations across sessions.

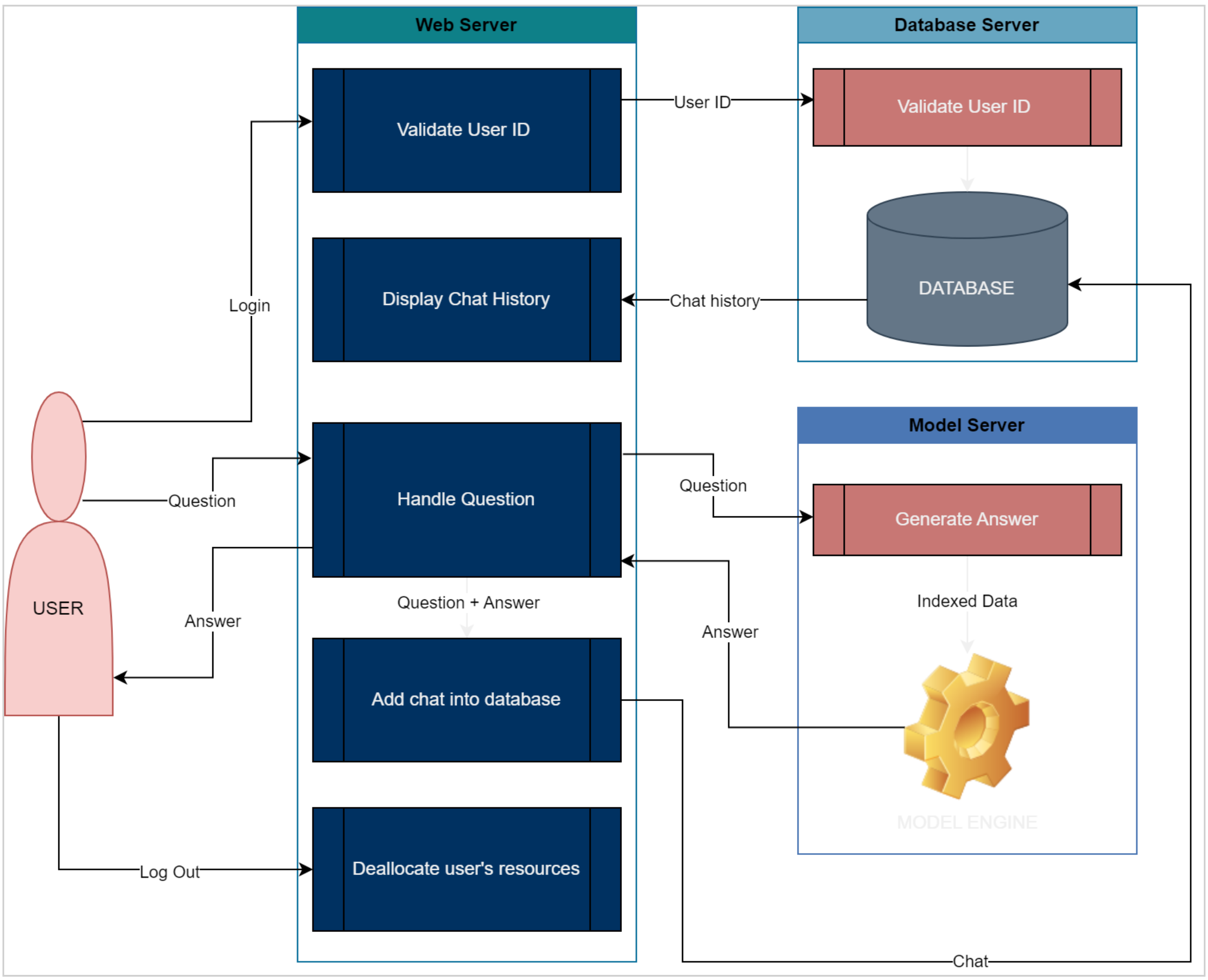
1.1.2.4 Heartbeat Server

- The Heartbeat Server is a service designed to monitor and manage the status of user sessions.
- Its primary responsibility is to ensure that user sessions are correctly initiated, utilized, and terminated.
- This component acts as a guardian for session integrity, preventing issues like unintended session prolongation or premature termination.
- It actively checks and manages the health of ongoing sessions, contributing to the reliability and stability of the overall system.

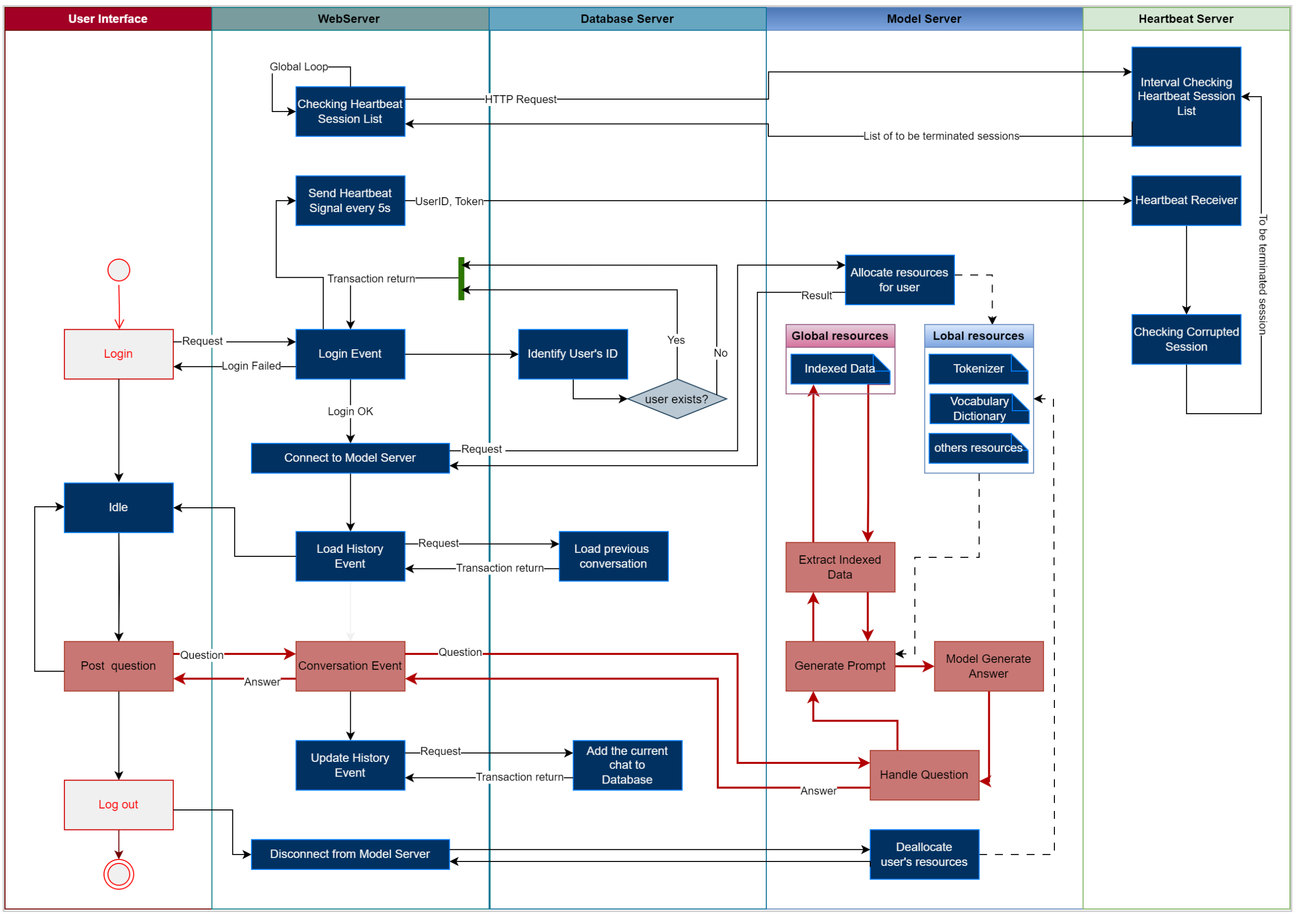
1.1.2.5 Document Server

- The Document Server serves as a local storage system dedicated to saving user-loaded documents.
- This component allows users to upload, store, and reference documents during their interactions with the chatbot.
- The Document Server supports the chatbot's ability to access and provide information from user-provided documents, enhancing the richness and breadth of the chatbot's knowledge base.
- It contributes to the system's versatility by enabling document-based interactions and responses.

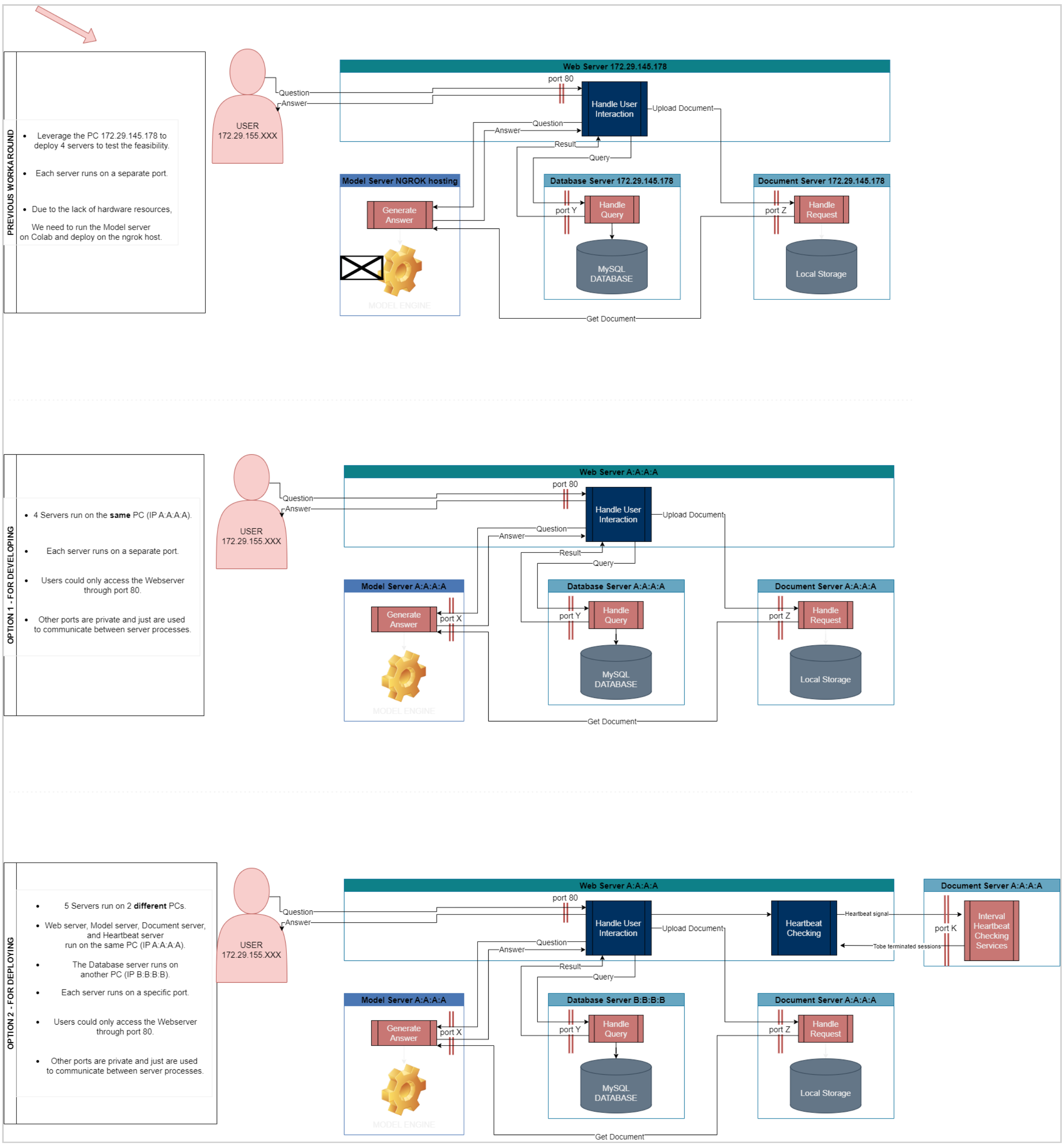
1.1.3 System Design User Viewpoint



1.1.4 System Design Developer Viewpoint



1.1.5 Network Design



PREVIOUS WORKAROUND

- Leverage the PC 172.29.145.178 to deploy 4 servers to test the feasibility.
- Each server runs on a separate port.
- Due to the lack of hardware resources, We need to run the Model server on Colab and deploy on the ngrok host.

OPTION 1 - FOR DEVELOPING

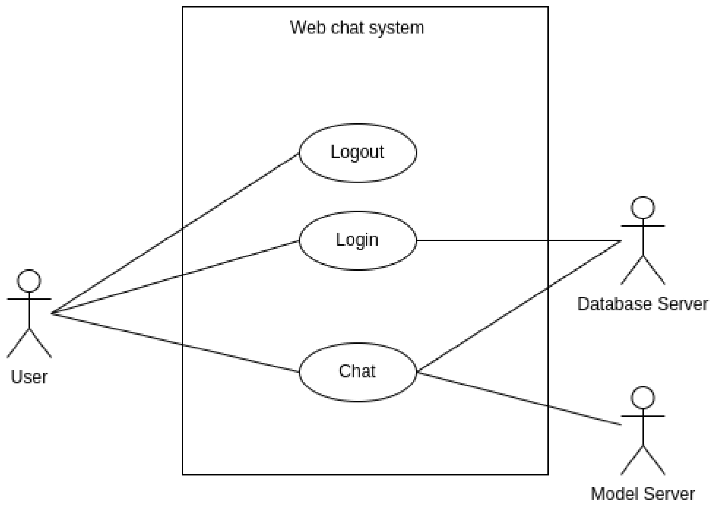
- 4 Servers run on the **same** PC (IP A:A:A:A).
- Each server runs on a separate port.
- Users could only access the Webserver through port 80.
- Other ports are private and just are used to communicate between server processes.

OPTION 2 - FOR DEPLOYING

- 5 Servers run on 2 **different** PCs.
- Web server, Model server, Document server, and Heartbeat server run on the same PC (IP A:A:A:A).
- The Database server runs on another PC (IP B:B:B:B).
- Each server runs on a specific port.
- Users could only access the Webserver through port 80.
- Other ports are private and just are used to communicate between server processes.

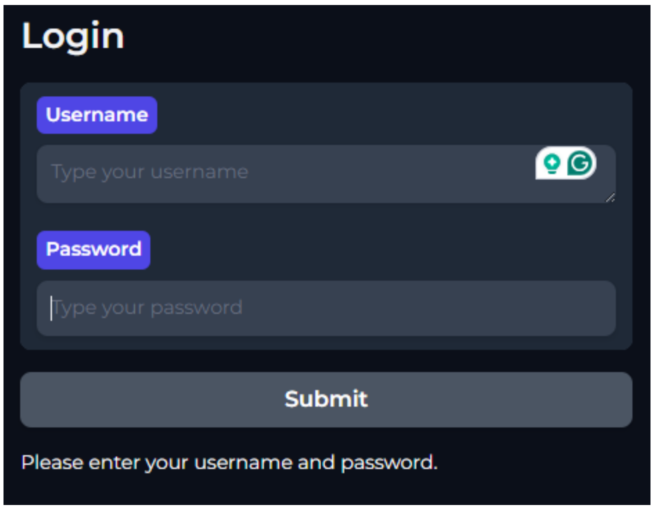
1.2 02_Web Chat Process Flow

1.2.1 Main features in the Chatbot WebUI



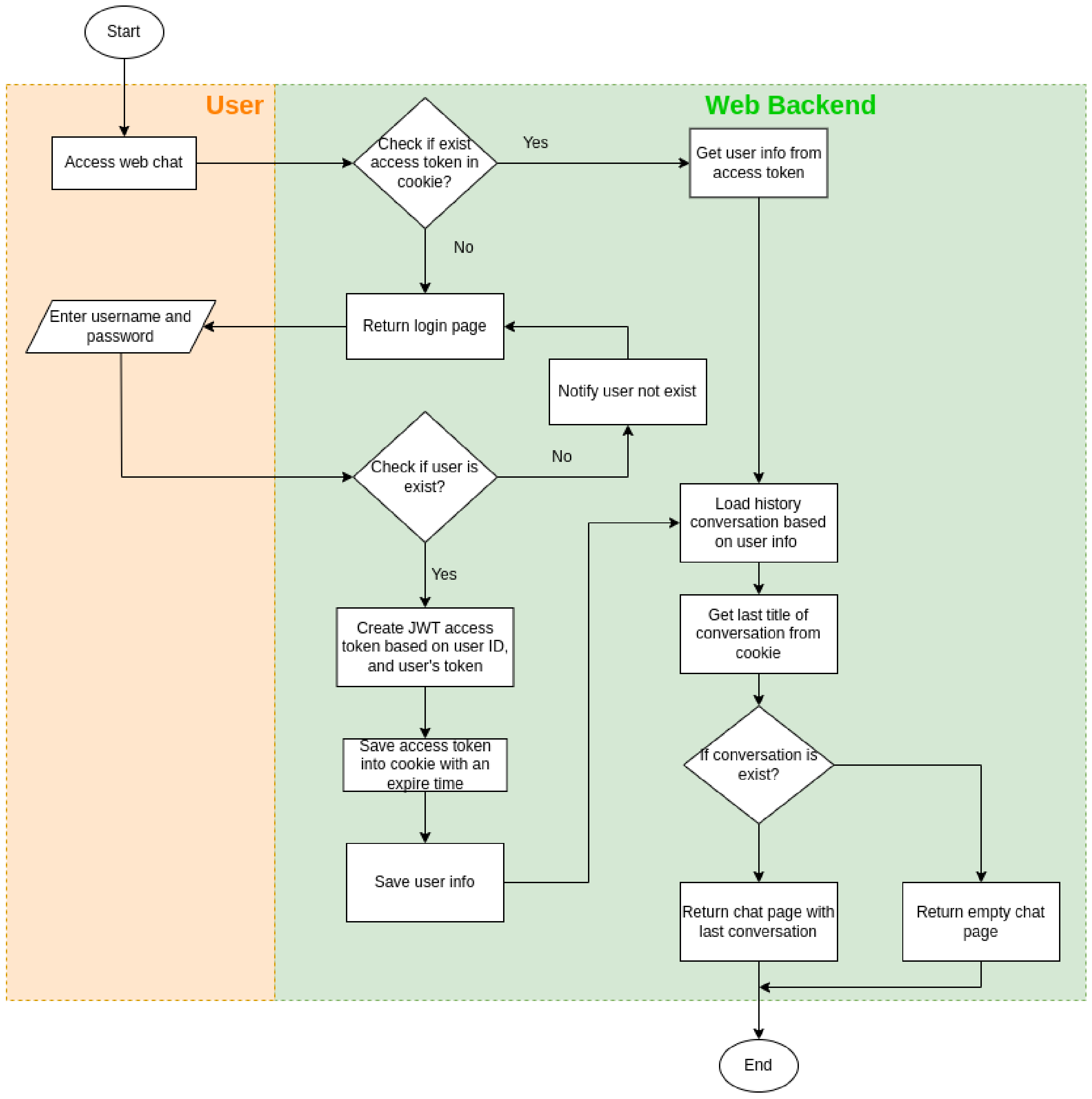
1.2.2 Login

1.2.2.1 UI



Please enter your username and password.

1.2.2.2 Backend flow


Start
User
Web Backend
Access web chat
Check if exist access token in cookie?
Yes
Get user info from access token
No
Enter username and password
Return login page
Notify user not exist
Check if user is exist?
No
Yes
Load history conversation based on user info
Create JWT access token based on user ID, and user's token
Get last title of conversation from cookie
Save access token into cookie with an expire time
If conversation is exist?
Save user info
Return chat page with last conversation
Return empty chat page
End

1.2.2.3 Backend Additional Service

Feature: User Access Limitation: This solution ensures that each user can only use one account to access the system at any given time.

=> **Heartbeat-checking solution**

1.2.2.3.1 On the heartbeat server side

The system initiates the process by receiving the sent heartbeats, including the username, from the web server when a user logs in.
Concurrently, an event loop checks if heartbeats are received from users within the defined interval.
If no heartbeat is received within the specified time, the username is added to a set.

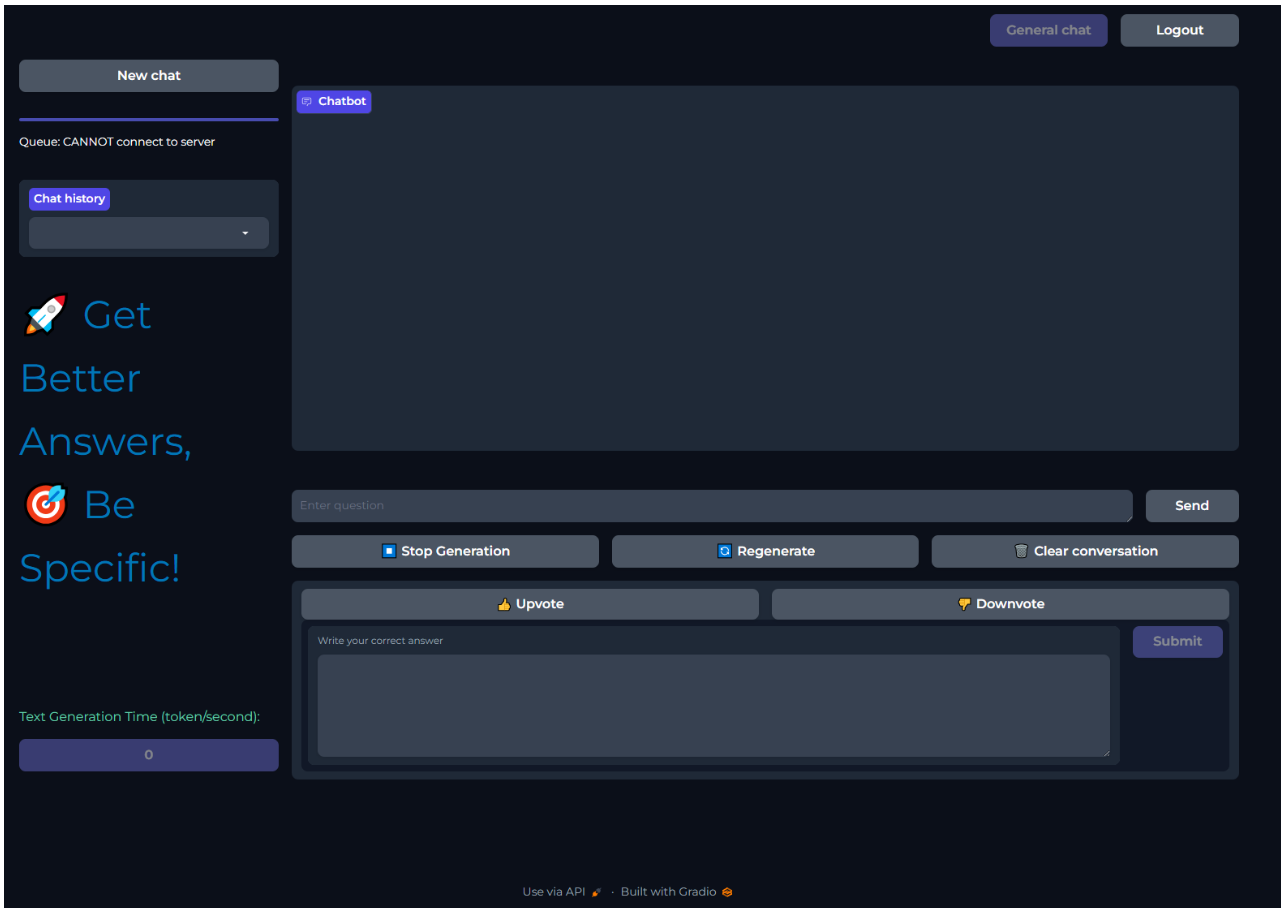
1.2.2.3.2 On the web server side

The main event loop sends a GET request to the heartbeat server after a defined interval.
The request queries the heartbeat server for any usernames that have lost connection.
If the response contains such names, the web server automatically logs out the affected account from the system.

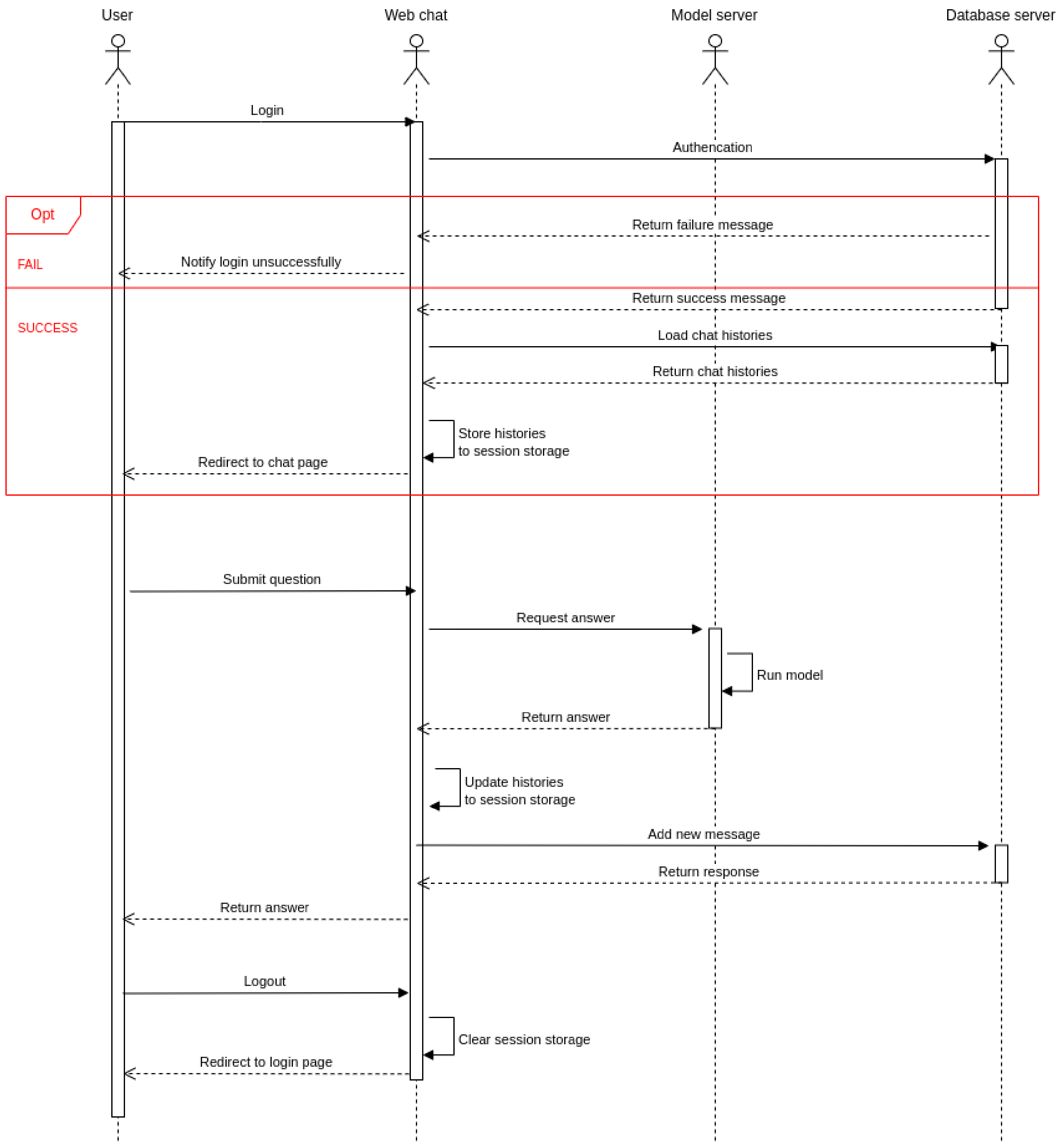
1.2.2.4 Login feature

- **Auto login**: When a user successfully login, I will store their access token in the browser's cookie. This access token has an expiration time, which is currently set to 60 seconds. When the user revisits the web chat, they will not need to log in again if the access token has not expired. The web chat will automatically redirect the user to the chat page.
- **Auto open last conversation**: when the user successfully logs in, the web chat will redirect to the chat page, and open the last conversation if it still exists in DB (Database).

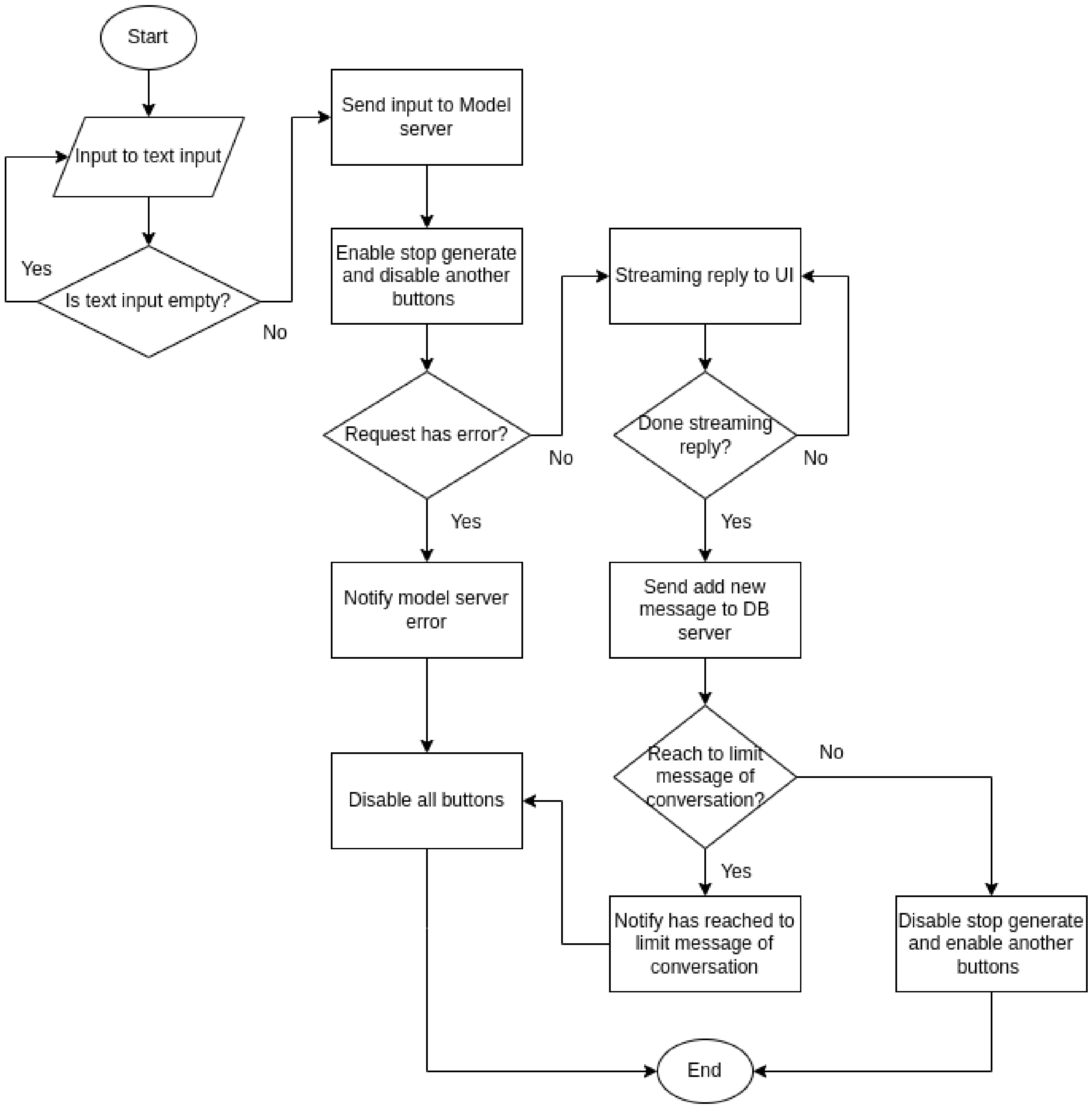
1.2.3 Main Chat UI


General chat
Logout
New chat
Queue: CANNOT connect to server
Chat history
🚀 Get Better Answers, 🎯 Be Specific!
Text Generation Time (token/second):
0
Chatbot
Enter question
Send
Stop Generation
Regenerate
Clear conversation
Upvote
Downvote
Write your correct answer
Submit
Use via API · Built with Gradio

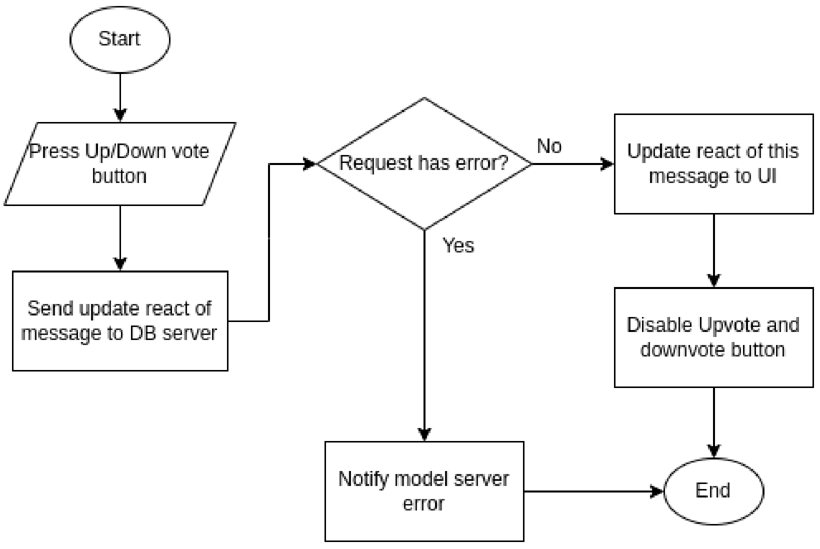
1.2.4 The sequence diagram describes the basic flow of the Chatbot



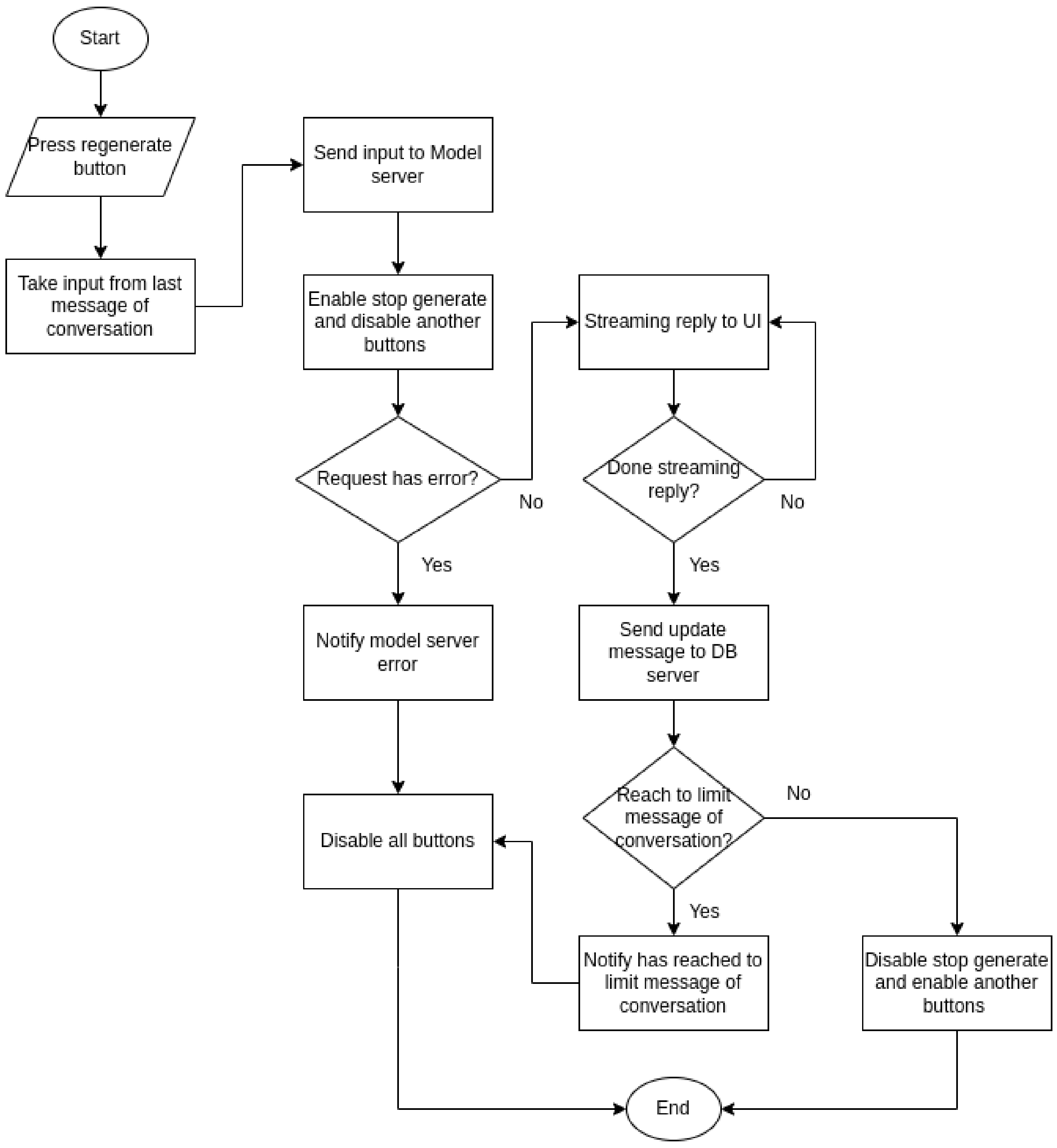
1.2.5 Chat flow


Start
Input to text input
Is text input empty?
Yes
No
Send input to Model server
Enable stop generate and disable another buttons
Request has error?
No
Yes
Streaming reply to UI
Done streaming reply?
No
Yes
Notify model server error
Send add new message to DB server
Reach to limit message of conversation?
No
Yes
Disable all buttons
Notify has reached to limit message of conversation
Disable stop generate and enable another buttons
End

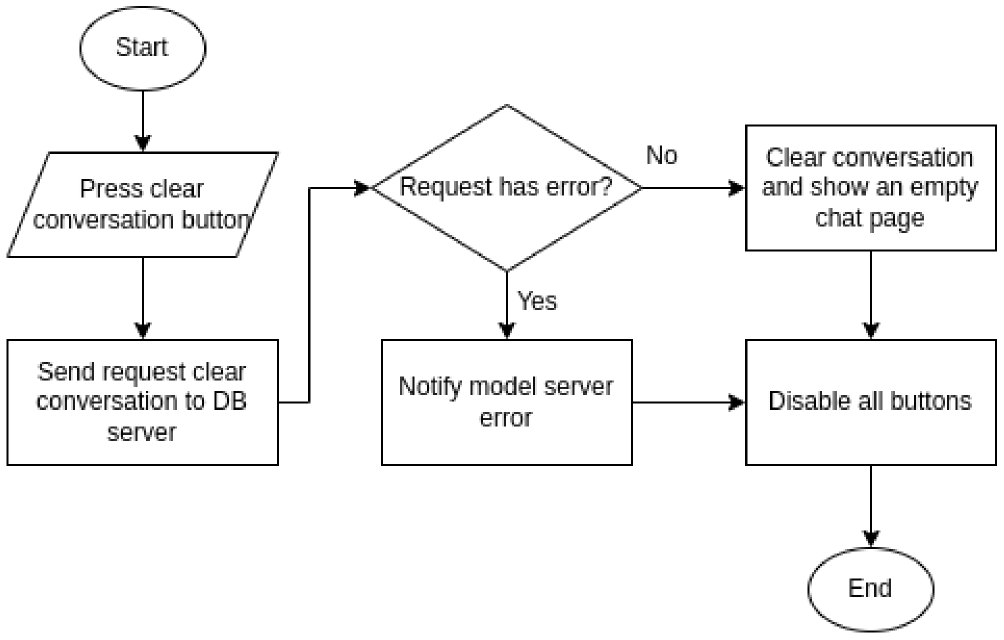
1.2.6 Flow chart of Up/down voting


Start
Press Up/Down vote button
Send update react of message to DB server
Request has error?
No
Yes
Update react of this message to UI
Disable Upvote and downvote button
Notify model server error
End

1.2.7 Flow chart of regeneration

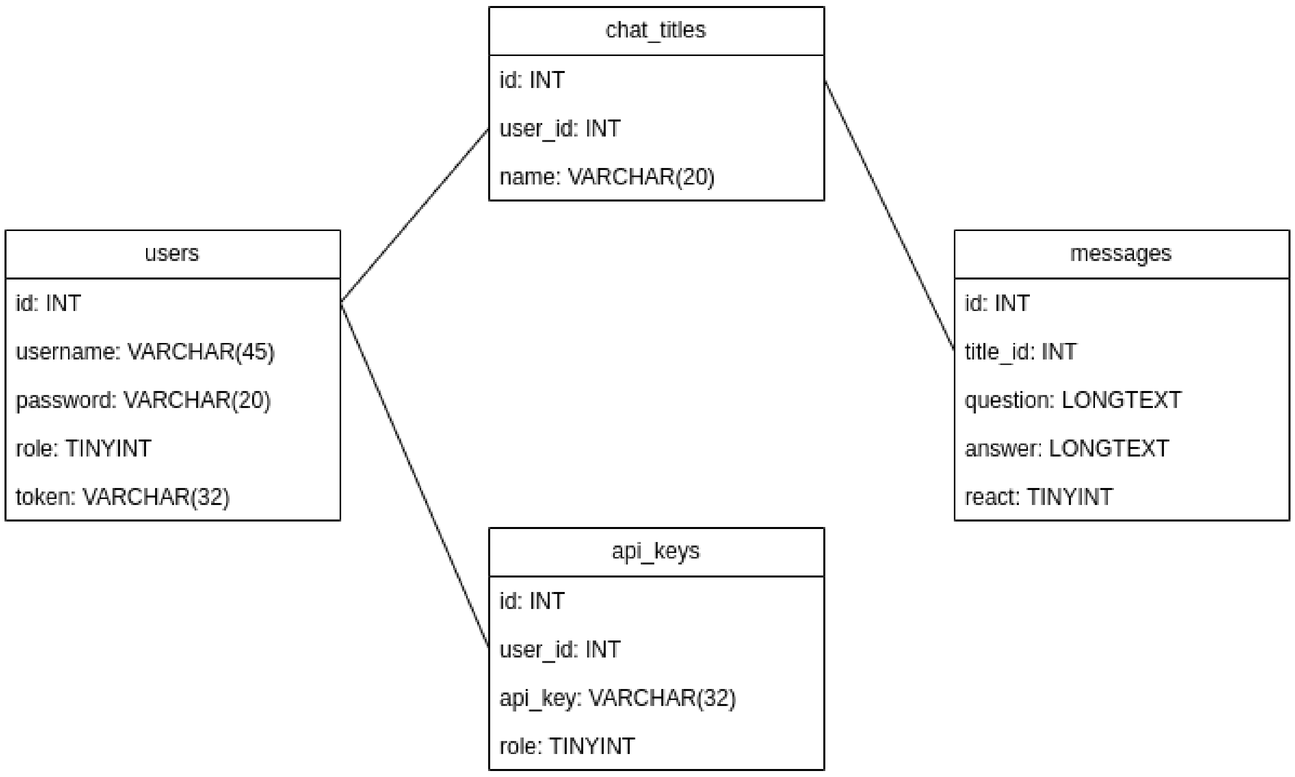

Start
Press regenerate button
Take input from last message of conversation
Send input to Model server
Enable stop generate and disable another buttons
Request has error?
No
Yes
Streaming reply to UI
Done streaming reply?
No
Yes
Notify model server error
Send update message to DB server
Disable all buttons
Reach to limit message of conversation?
No
Yes
Notify has reached to limit message of conversation
Disable stop generate and enable another buttons
End

1.2.8 Flow chart of conversation clearing


Start
Press clear conversation button
Send request clear conversation to DB server
Request has error?
No
Yes
Clear conversation and show an empty chat page
Notify model server error
Disable all buttons
End

1.3 03_Database Server

1.3.1 Database organization



- **users:** Store information of users
  - id: identification number of user
  - username: user name of user
  - password: password of the user
  - role: decentralization of access resources of the user
  - token: for security purposes
- **chat_titles**: Store identification of conversations
  - id: identification number of conversation
  - user_id: identification number of the user that the conversation belongs to
  - name: name of conversation
- **messages**: Store a pair of questions and answer
  - id: identification of the pair
  - title_id: identification number of conversation that this pair belongs to
  - question: the question of the pair
  - answer: answer of the pair

  - react: reaction of the user with the answer of this pair
- **api_keys**: store the public key of the user. Users can use this key to directly connect to the model (Like the OpenAI API key)
  - id: identification of API key
  - user_id: identification number of the user that the API key belongs to
  - api_key: value of API key
  - role: to determine the role of this API key

1.3.2 Choosing a Database Management System

1.3.2.1 Comparison table

DBMS	**Summaries**	**Pros**	**Cons**
MySQL	- A popular open-source relational database management system known for its fast performance, scalability, and reliability. Widely used for web applications.	- High performance and scalability. - Large community support and a wide range of resources available. - Cross-platform compatibility. - Flexible and can be used with multiple programming languages.	- Limited support for unstructured data. - Lacks some advanced features available in other DBMSs. - Can be difficult to optimize and fine-tune for specific workloads.
PostgreSQL	- An open-source relational database management system known for its robustness, reliability, and advanced features such as support for geographic data and JSON data types.	- Robust and reliable with high data integrity and security features. - Supports a wide range of data types and offers advanced features like geographic data and JSON data types. - Large community support and a wide range of resources available. - Cross-platform compatibility.	- Performance can be slower compared to other DBMS, especially under high load. - Some advanced features can be complex to use and may require additional setup and configuration. - Limited commercial support compared to proprietary DBMS.

DBMS	Summaries	Pros	Cons
Oracle	- A proprietary relational database management system known for its scalability, reliability, and advanced features such as support for partitioning and clustering.	- Highly scalable and reliable with advanced features like partitioning and clustering. - Excellent support for large datasets and complex data structures. - Offers a range of data security and management features. - Comprehensive documentation and commercial support.	- Proprietary license and higher cost compared to other DBMS. - Can be complex to set up and manage. - Requires dedicated hardware resources for optimal performance.
MongoDB	- A popular NoSQL database management system known for its flexibility, scalability, and ability to handle unstructured data. Widely used for big data applications	- Flexible and scalable with support for unstructured data. - High performance and can handle large volumes of data. - Easy to set up and use with a straightforward document model. - Excellent support for distributed database systems.	- Limited support for transactions and ACID compliance. - May not be ideal for applications that require complex querying or reporting. - No built-in support for joins, which may require additional processing.
SQLite	- A lightweight, embedded relational database management system known for its simplicity, small size, and easy integration into other applications. Often used for mobile applications and small web applications.	- Lightweight and easy to use, with a small footprint. - Fast performance and can handle small to medium-sized databases. - Cross-platform compatibility. - Offers a range of features for embedded systems.	- Limited scalability and performance issues with larger datasets. - Not ideal for complex or highly concurrent applications. - No built-in support for network-based access, requiring additional setup for multi-user environments.

1.3.2.2 Conclusion

I will pick up a DBMS that is popular, easy to use, and can support scalability, so the most suitable for that is **MySQL**

1.3.2.3 Here are some more detailed advantages of MySQL

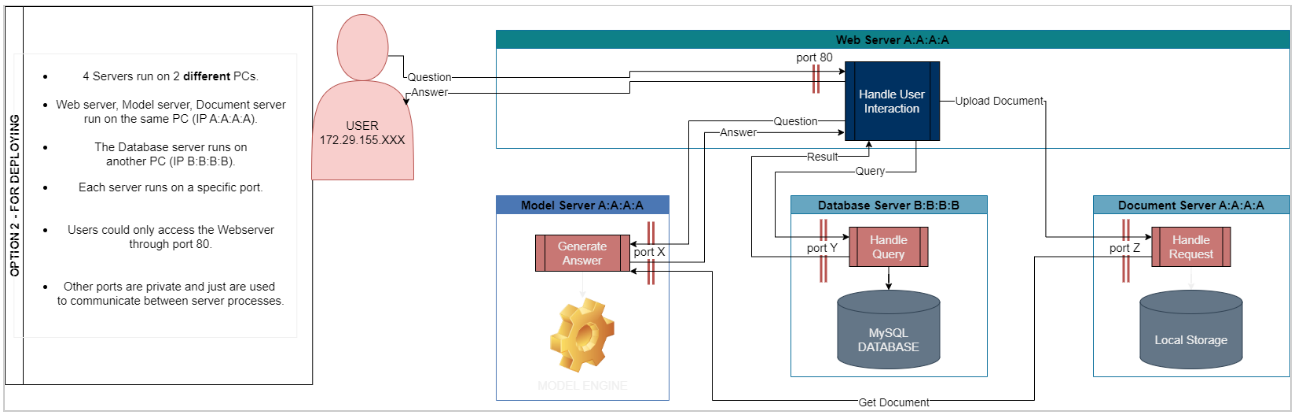
- **Open-source** and **Cost-effective**: MySQL is an open-source DBMS, meaning it is freely available for use and can be modified to suit specific needs. This makes it a cost-effective option, especially for small to medium-sized projects or startups with limited budgets.

- **Wide Adoption and Community Support**: MySQL has been around for over two decades and has gained significant popularity. It has a large user base and an active community of developers and contributors. This means you can find extensive documentation, tutorials, and support forums to help you with any issues you might encounter.
- **High Performance**: MySQL is known for its high-performance capabilities, particularly in read-intensive workloads. It is optimized for quick data retrieval and can handle a large number of concurrent connections efficiently. Additionally, MySQL offers various indexing techniques, caching mechanisms, and query optimization features to enhance performance.
- **Scalability and Flexibility:** MySQL is highly scalable and can handle large amounts of data and high traffic volumes. It supports replication, clustering, and sharding techniques, allowing you to distribute your database across multiple servers and achieve horizontal scalability. This flexibility makes it suitable for both small-scale applications and large enterprise-level systems.
- **Wide Range of Applications**: MySQL is a versatile DBMS that supports various types of applications, from simple websites and blogs to complex enterprise solutions. It offers robust support for different data types, including text, numeric, date/time, and spatial data. Furthermore, MySQL supports multiple programming languages and provides APIs for easy integration with different platforms and frameworks.
- **Security Features**: MySQL provides several security features to protect your data. It supports authentication and access control mechanisms, allowing you to define user privileges and restrict unauthorized access. MySQL also supports encrypted connections, and data encryption at rest, and has mechanisms for securing sensitive data.
- **Stability and Reliability**: MySQL is a mature and stable database system that has been extensively tested and improved over the years. It has a reputation for being reliable and is used by many large-scale applications that require high availability and data integrity.

2 02_Getting Started

2.1 01_Set Up Environment Instruction

AI Team has deployed a completed system as below.


OPTION 2 - FOR DEPLOYING

- 4 Servers run on 2 **different** PCs.
- Web server, Model server, Document server run on the same PC (IP A:A:A:A).
- The Database server runs on another PC (IP B:B:B:B).
- Each server runs on a specific port.
- Users could only access the Webserver through port 80.
- Other ports are private and just are used to communicate between server processes.

2.1.1 Get Resources

Clone the below repository, which contains the source code of all module servers.

Source code: https://gitlab.rvc.renesas.com/mcu-software/rh-autosar-software-2/mcal-guru.git

2.1.2 Setup Model Server, Web Server, Document Server, and Heartbeat Server

2.1.2.1 Prepare environment

- Prepare a powerful PC with GPUs for deploying servers.
- Request IT to install the following items:
    - Python3.11.4 and Pip.
    - Cuda==11.8
    - Pytorch==2.0.1+cu118.
    - VSCode.

2.1.2.2 Prepare source code

- Clone the repository from this URL: https://gitlab.rvc.renesas.com/mcu-software/rh-autosar-software-2/mcal-guru.git

2.1.2.3 Prepare workspace

2.1.2.3.1 Install Python packages

- Create virtual environment

```
python -m venv ./venv
cd venv
cd Scripts
activate
python -m pip install --upgrade pip
```

- Install packages and dependencies

```
pip install -r requirements.txt
```

2.1.2.3.2 Install auto-gptq

- On the local PC run

```
https://github.com/PanQiWei/AutoGPTQ/releases/download/v0.4.2/auto_gptq-0.4.2+cu118-cp311-cp311-win_amd64.whl
```

- Copy the wheel to the sharing device (accessible from the Lab PC).
- Activate the virtual environment.
- Run this command:

```
pip install auto_gptq-0.4.2+cu118-cp311-cp311-win_amd64.whl
```

2.1.2.3.3 Install Dependencies

2.1.2.3.3.1 LLM Model

Need to prepare 2 huggingface models to folder './model':

- OpenOrca-Platypus2-13B-GPTQ
- all-mpnet-base-v2-table

```
Copy from the shared network device DTV(\\rvc-vnas-01)(V:)
\Users\hyla\AI_MCAL\Source_Code\model
```

2.1.2.3.3.2 NLTK Vocabularies

- corpora
- taggers
- tokenizers

```
Copy from the shared network device MobAP(\\rvc-vnas-01)(M:)
\u\hyla\AI_CHATBOT\resources\nltk_data
```

```
Paste into C:\Users\rvc_sw_mss1_common\AppData\Roaming\
```

2.1.2.3.3.3 Spacy library

- en_core_web_lg
- en_core_web_lg-3.5.0.dist-info
- en_core_web_sm
- en_core_web_sm-3.5.0.dist-info

```
Copy from the shared network device MobAP(\\rvc-vnas-01)(M:)
\u\hyla\AI_CHATBOT\resources\spaCy
```

```
Paste into venv\Lib\site-packages\
```

2.1.3 Setup Database Server

2.1.3.1 Prepare environment

- Prepare a PC for deploying the database server.

- Request IT to install the following items:
    - MySQL Server on this PC.
    - Python3.11.4 and Pip.
    - Config a fixed IP address for the database PC.
    - MySQL Workbench on your developing PC (Workbench is used for monitoring and modifying the database from other PCs).
- Request Network Team:
    - Grand Permission to access the database PC.

2.1.3.2 Setup Database

- After the installation of the MySQL Server, we now can connect to it. There are 3 methods for connecting: Command-line, Using MySQL Workbench, and Script.
- But for easy to use and intuitive we use MySQL Workbench for managing DB.
- Get database schema from this URL: https://gitlab.rvc.renesas.com/mcu-software/rh-autosar-software-2/mcal-guru/-/blob/main/MCAL_BOT_Chatbot/MCAL_BOT.db
- Create a new Schema and import the downloaded schema **MCAL_Bot.sql** to the schema.

2.1.3.3 Prepare source code

- Get all files from this URL: https://gitlab.rvc.renesas.com/mcu-software/rh-autosar-software-2/mcal-guru/-/tree/main/MCAL_BOT_Chatbot/DBServer
- Get the requirement file from: https://gitlab.rvc.renesas.com/mcu-software/rh-autosar-software-2/mcal-guru/-/blob/main/MCAL_BOT_Chatbot/requirements.txt

2.1.3.4 Prepare workspace

- Create virtual environment

```
python -m venv ./venv
cd venv
cd Scripts
activate
python -m pip install --upgrade pip
```

- Install packages and dependencies

```
pip install -r requirements.txt
```

2.1.3.5 Database Server Appendix

2.1.3.5.1 Useful commands

2.1.3.5.1.1 Start MySQL service

```
sudo service mysql start
```

2.1.3.5.1.2 Check MySQL Service Status

Check MySQL Service Status:

```
systemctl status mysql
```

Check MySQL Process

```
ps aux | grep mysql
```

Check MySQL Port

```
sudo ss -tuln | grep 3306
```

2.1.3.5.1.3 Identify access to the MySQL server

```
sudo tail /var/log/mysql/error.log
```

2.1.3.5.1.4 Convert a MySQL database to an SQL file

```
mysqldump -u [username] -p [password] [database_name] > [output_file.sql]
```

2.1.3.5.2 Detailed guidelines

2.1.3.5.2.1 Create a new database

To create a new database using an SQL script in MySQL Workbench, you can follow these steps:

1. Connect to the MySQL Server:
   - Open MySQL Workbench and connect to your MySQL server as described earlier.
2. Create a New Schema:
   - In the left sidebar under the "SCHEMAS" tab, right-click and select "Create Schema."
   - Enter a name for the new schema and click "Apply."
3. Open the SQL Script:
   - Open the SQL script file that contains the database creation statements. You can do this by clicking on the "File" menu and selecting "Open SQL Script," and then browsing to the location of the script file.
4. Execute the SQL Script:
   - Once the SQL script is open, click on the "Query" tab at the top to open the SQL editor.
   - Make sure that the current schema in the SQL editor is set to the newly created schema (you can change it from the dropdown).
   - Select the entire SQL script in the editor (use `Ctrl+A`), or select specific parts of the script that you want to execute.
   - Click the lightning bolt icon or press `Ctrl+Enter` to execute the selected SQL script.
   - The script will run, and the database objects will be created in the selected schema.
5. Verify the Database Objects:
   - After executing the SQL script, you can verify the newly created database objects (tables, views, etc.) by refreshing the schema in the left sidebar.
6. Use the Database:
   - You can now use the newly created database to perform data operations, create tables, insert data, etc. by writing SQL queries in the SQL editor.

2.1.3.5.2.2 Create an ERD and visualize your table

Here's a detailed guide to using the "Data Modeling" tool in MySQL Workbench to create an ERD and visualize your tables:

1. Open MySQL Workbench and Connect to the Server:
   - Open MySQL Workbench and establish a connection to your MySQL server by clicking on the "+" icon in the "MySQL Connections" section.
   - Enter the necessary connection details like hostname, port, username, and password.

2. Open the "Data Modeling" Tool:
    - Once you are connected to the MySQL server, go to the top menu and click on "Database" > "Reverse Engineer" or "Create EER Model."
3. Reverse Engineer or Create a New Model:
    - If you already have tables in your schema, choose "Reverse Engineer" to import the existing tables into the "Data Modeling" tool.
    - If you want to create a new model from scratch, choose "Create EER Model."
4. Import Existing Tables (if applicable):
    - If you select "Reverse Engineer," MySQL Workbench will display a list of available schemas. Choose the schema containing your tables (e.g., "chatbot_db") and click "Next."
    - Select the tables you want to import into the model and click "Execute." The selected tables will now be added to the "Data Modeling" tool.
5. Design the ERD:
    - In the "Data Modeling" tool, you'll see a graphical canvas where you can design your ERD.
    - To add tables to the canvas, click on the "Add Table" icon (the first icon on the left toolbar) and specify the table name and its columns.
6. Define Relationships (if applicable):
    - To define relationships between tables, click on the "Add Relationship" icon (the second icon on the left toolbar) and connect the primary key of one table to the foreign key of another.
7. Customize ERD Layout (Optional):
    - You can drag and drop tables and relationships to arrange them on the canvas as you like.
    - Use the options in the toolbar to customize the appearance of tables, columns, and relationships.
8. Save the Model:
    - Once you have designed your ERD, save the model by clicking "File" > "Save Model" and give it a name (e.g., "chatbot_db_model.mwb").
9. Generate SQL Script (Optional):
    - If you plan to execute the model to create the database schema, you can click "File" > "Export" > "Forward Engineer SQL CREATE Script." This will generate the SQL script for creating the schema based on your ERD.

2.1.3.5.2.3 View or edit the table data in MySQL Workbench

Method 1: Using SQL Editor

1. Connect to your MySQL server and select the desired schema (database) from the "SCHEMAS" panel on the left.
2. In the "Navigator" panel on the left, click on the "+" icon next to the schema to expand it and view its tables.

3. Right-click on the table whose data you want to view or edit and select "Send to SQL Editor" > "Select Rows - Limit 1000."
4. The "SQL Editor" panel will open with a SELECT query that retrieves the first 1000 rows from the selected table. You can modify the query as needed.
5. To view the data, click on the lightning bolt icon or press `Ctrl+Enter` (or `Cmd+Enter` on macOS) to execute the query. The results will be displayed in the "Result Grid" below.
6. To edit the data, double-click on a cell in the "Result Grid" to enter edit mode. Make the desired changes and press `Ctrl+Enter` (or `Cmd+Enter` on macOS) to apply the changes to the database.

Method 2: Using Table Data

1. Connect to your MySQL server and select the desired schema (database) from the "SCHEMAS" panel on the left.
2. In the "Navigator" panel on the left, click on the "+" icon next to the schema to expand it and view its tables.
3. Right-click on the table whose data you want to view or edit and select "Table Data."
4. The "Table Data" panel will open, displaying the contents of the selected table.
5. To view or edit the data, click on the cells directly in the table to make changes.
6. To apply changes, click on the "Apply" button in the toolbar above the table.
7. To discard changes without saving, click on the "Revert" button.

2.2 02_Deployment Instruction

2.2.1 Overview

2.2.1.1 Component Review

1. Database Server (Flask): This server handles user information, login, and logout processes. It is responsible for managing user sessions and authentication. Flask is a suitable choice for this task, as it is a lightweight and easy-to-use framework for building web applications.
2. Model Server (Flask): The model server runs on Flask and hosts the language model for generating text. It creates a session for each user upon login and streams text to the client when requested.
3. Web UI (Gradio): The Web UI built on Gradio will interact with both the Database Server (for login/logout) and the Model Server (for generating text).

2.2.1.2 Regarding multi-threading and multi-processing support

- Flask has built-in support for handling multiple requests concurrently.
- Gradio is designed to work with any Python web framework, including Flask. It leverages the capabilities of the underlying framework to handle multiple requests. Flask servers are capable of handling multiple users, and Gradio will be able to interact with them concurrently.

2.2.2 Deployment Solution

1. Prepare local PC: Make sure the local PC is connected to the local network and has a static IP address or a hostname that internal users can use to access the Web UI.
2. Run the Gradio Web UI: Run the Gradio Web UI application using the Python script containing the source code. This will start the Gradio server on your local PC and make it ready to accept incoming requests.
3. Configure Firewall and Network Settings: Ensure that the local PC's firewall allows incoming traffic on the port where the Gradio Web UI is running (e.g., port 80 or 8080). Also, configure the local network settings to forward incoming requests to the local PC's IP address or hostname.
4. Share the Access URL: Once the Gradio Web UI is running and accessible on the local PC, share the access URL (IP address or hostname) with internal users. They can use this URL to access the Web UI from their web browsers.
5. Handle Multiple Users: Since Gradio automatically handles multiple users and concurrency, we don't need to create separate instances or processes for each user. Gradio will manage the user interactions and ensure that each user's requests are processed independently.
6. Consider Security: Keep in mind that the Gradio Web UI will be accessible on the local network, so ensure that proper security measures are in place. Use strong passwords, configure access control, and regularly update your software to protect against potential security threats.
7. Monitor and Maintain: Once the Gradio Web UI is deployed, monitor its performance and user feedback. If needed, can make improvements or updates to enhance the user experience.

2.2.3 Deployment Instruction

The maintainer should start each module in this order sequentially: Database Server → Document Server → Heartbeat Server → Model Server → Web Chat.

2.2.3.1 Database Server

- In the database server PC, open the database server script at **DBserver/mysqlServer.py.**
- Before running, you need to config the database connection following the information you have set when installing the MySQL server.

```
mydb = mysql.connector.connect(
    host="127.0.0.1",
    user="root",
    passwd="password",
    database="ChatBot"
)
```

- In the source code workspace, open the terminal (Window → Type "cmd" → Enter).
- Activate Virtual Environment.

```
cd venv
cd Scripts
activate
```

- Run script

```
cd DBServer
python mysqlServer.py
```

2.2.3.2 Document Server

- Implemented in **./mcal-guru/MCAL_BOT_Chatbot/DBServer/documentServer.py** script
- Before running you need to configure the path to the folder store file document. You can config it in the file **./mcal-guru/MCAL_BOT_Chatbot/config.json**

```
{
    ...,
    "path_to_database" : "C://Users//rvc_sw_mss1_common//Desktop//test//
Document_Database//",
    ...
}
```

- The DataBase folder contains 2 folders **internal** (Storing document default in the system) and **external** (Storing document uploaded by user).
- In the source code workspace, open the terminal (Window → Type "cmd" → Enter).
- Activate Virtual Environment

```
cd venv
cd Scripts
activate
```

- Run script:

```
cd DBServer
python documentServer.py
```

2.2.3.3 Heartbeat Server

- Implemented in **./mcal-guru/MCAL_BOT_Chatbot/DBServer/heartbeats_server.py** script
- In the source code workspace, open the terminal (Window → Type "cmd" → Enter).
- Activate Virtual Environment

```
cd venv
cd Scripts
activate
```

- Run script

```
cd DBServer
python heartbeats_server.py
```

2.2.3.4 Model Server

- The model server is implemented in a Jupyter Notebook.
- Located at **./mcal-guru/MCAL_BOT_Chatbot/ModelServer/MCAL_BOT_FlaskAPI.ipynb** notebook.
- In the source code workspace, open the terminal (Window → Type "cmd" → Enter).
- Activate Virtual Environment

```
cd venv
cd Scripts
activate
```

- Run the Jupyter Notebook server.

```
jupyter notebook --ip <IP address of the server PC> --port <port number>
```

- Open the Model Server notebook.
- In the tool taskbar, click "Cell" → "Run All"

2.2.3.5 Web Server

- Program in **./mcal-guru/MCAL_BOT_Chatbot/WebChat/main.py** script
- Config some parameters in **./mcal-guru/MCAL_BOT_Chatbot/config.json**

```
{
    "document_server_address": "http://127.0.0.1:2234/",
    "model_server_address": "https://172.29.173.29:8000",
    "mysql_server_address" : "http://172.29.173.72:1234",
    "path_to_database" : "C://Users//rvc_sw_mss1_common//Desktop//test//
Document_Database//",
    "cookie_expire_time_second": 60,
    "max_number_chat": 15,
    "max_number_chat_file_chat_mode": 15
}
```

- To deploy the Gradio Web UI on a PC for internal users to access, just configure the WebUI. Gradio supports handling multiple concurrent connections and provides better performance than the built-in development server.

```
webchat.launch(share=True, max_threads=50, auth_message="Please login to
use the service", inbrowser=False, server_name='172.29.173.88',
server_port=8000)
```

- In the source code workspace, open the terminal (Window → Type "cmd" → Enter).
- Activate Virtual Environment

```
cd venv
cd Scripts
activate
```

- Run script:

```
cd WebChat
python main.py
```

Access the Web UI: Once WebUI is running, your Gradio Web UI will be accessible at 172.29.173.88:8000 in the browser. Users can access it using the local IP address or hostname of their PC within the internal network.

3 03_AI Services Design

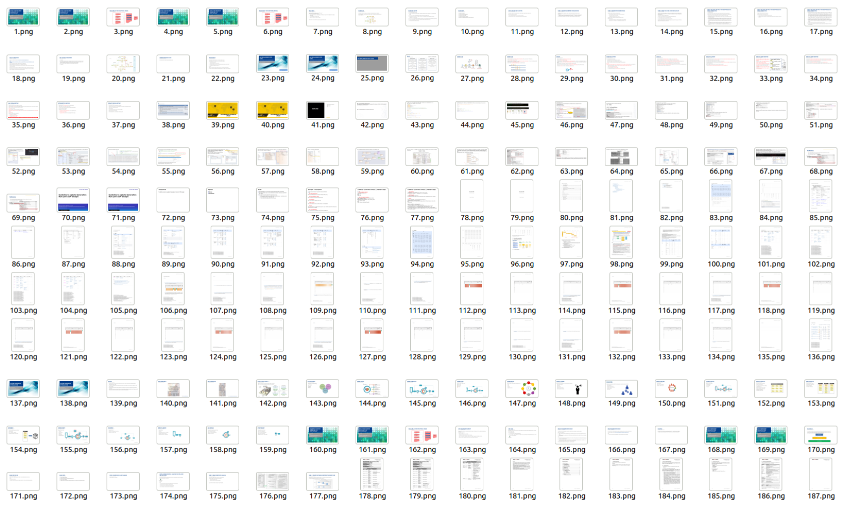
3.1 [AI - NLP] Detect Figure in Document

3.1.1 General

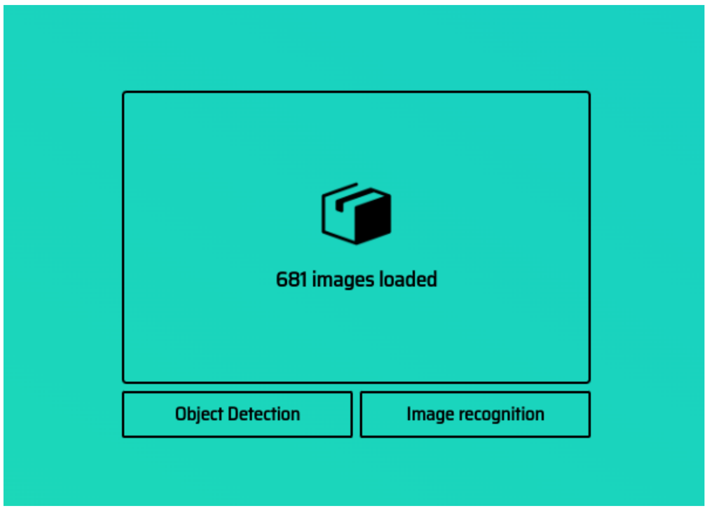
In document it have many figure which contain helpful information. But when i using python script convert pdf document to html it can not extract fully figure. So i using a object detection model to detect figure.

3.1.2 Data collection

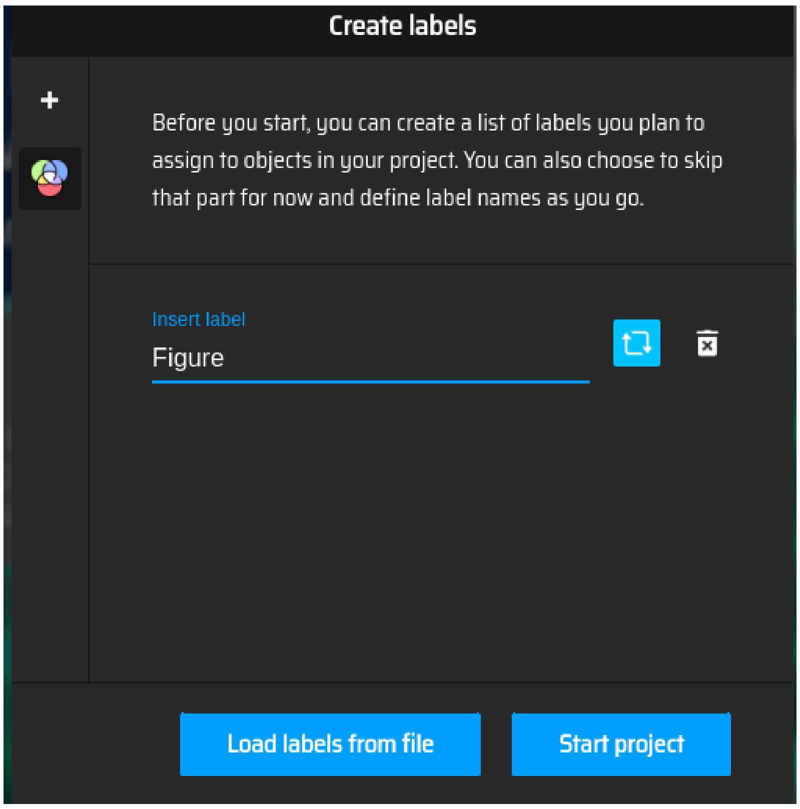
I have almost 700 images cutting directly from the document


1.png 2.png 3.png 4.png 5.png 6.png 7.png 8.png 9.png 10.png 11.png 12.png 13.png 14.png 15.png 16.png 17.png
18.png 19.png 20.png 21.png 22.png 23.png 24.png 25.png 26.png 27.png 28.png 29.png 30.png 31.png 32.png 33.png 34.png
35.png 36.png 37.png 38.png 39.png 40.png 41.png 42.png 43.png 44.png 45.png 46.png 47.png 48.png 49.png 50.png 51.png
52.png 53.png 54.png 55.png 56.png 57.png 58.png 59.png 60.png 61.png 62.png 63.png 64.png 65.png 66.png 67.png 68.png
69.png 70.png 71.png 72.png 73.png 74.png 75.png 76.png 77.png 78.png 79.png 80.png 81.png 82.png 83.png 84.png 85.png
86.png 87.png 88.png 89.png 90.png 91.png 92.png 93.png 94.png 95.png 96.png 97.png 98.png 99.png 100.png 101.png 102.png
103.png 104.png 105.png 106.png 107.png 108.png 109.png 110.png 111.png 112.png 113.png 114.png 115.png 116.png 117.png 118.png 119.png
120.png 121.png 122.png 123.png 124.png 125.png 126.png 127.png 128.png 129.png 130.png 131.png 132.png 133.png 134.png 135.png 136.png
137.png 138.png 139.png 140.png 141.png 142.png 143.png 144.png 145.png 146.png 147.png 148.png 149.png 150.png 151.png 152.png 153.png
154.png 155.png 156.png 157.png 158.png 159.png 160.png 161.png 162.png 163.png 164.png 165.png 166.png 167.png 168.png 169.png 170.png
171.png 172.png 173.png 174.png 175.png 176.png 177.png 178.png 179.png 180.png 181.png 182.png 183.png 184.png 185.png 186.png 187.png

After that i using https://www.makesense.ai/ to label image. After upload image choose object detection



Create a label name Figure



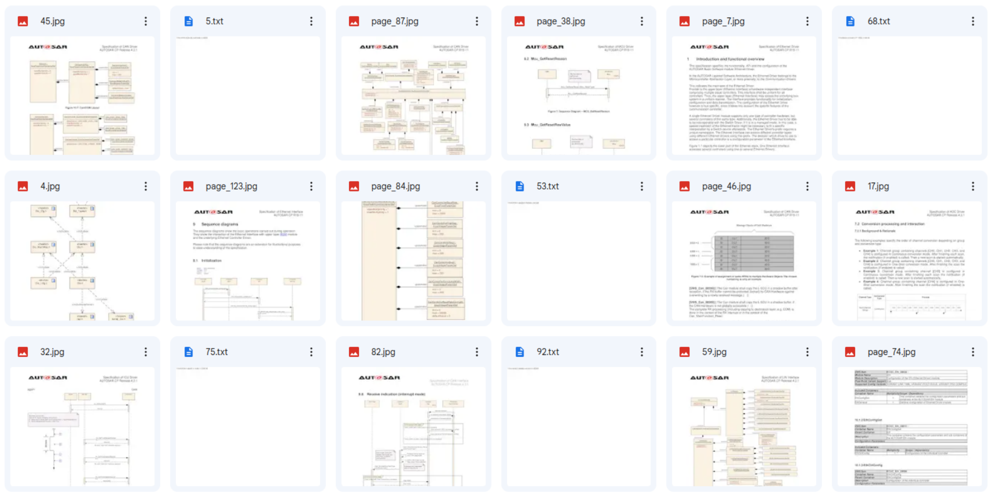
Create labels

Before you start, you can create a list of labels you plan to assign to objects in your project. You can also choose to skip that part for now and define label names as you go.

Insert label: Figure

And then click Start project.

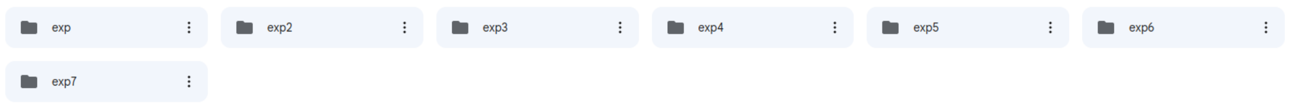
After label you will have text file correspond to each image



3.1.3 Training model

Run this collab https://colab.research.google.com/drive/1GmBd-H6QX_YBVwabUQiDybYUle9wAhR3?usp=sharing

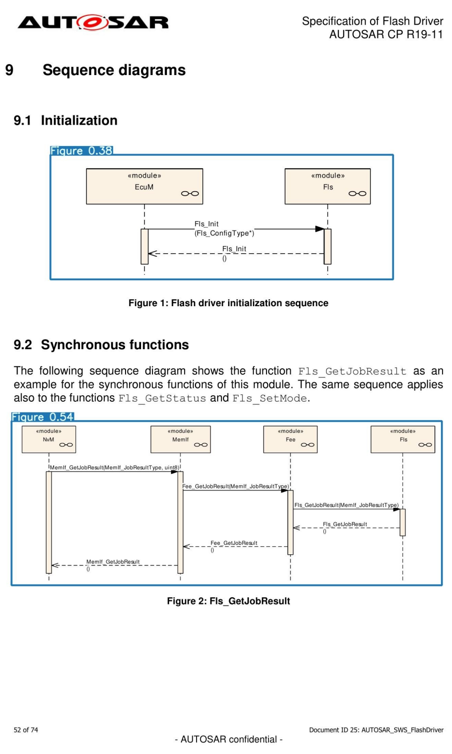
After training weight file will save in Yolov7/Runs/Train/Exp



in this case I train 7 times so that i have 7 folder. You need download folder and run model by this script : Detect Figure[1]

Result:

1 https://renesasgroup.sharepoint.com/:f:/r/sites/team_MCAL_CICD_QI_AI/Shared%20Documents/Resources/03_AI/04_Convert_Script/Detect%20Figure?csf=1&web=1&e=w5a7eL



Specification of Flash Driver
AUTOSAR CP R19-11

9 Sequence diagrams

9.1 Initialization

Figure 0.38
«module»
EcuM
«module»
Fls
Fls_Init
(Fls_ConfigType*)
Fls_Init
()

Figure 1: Flash driver initialization sequence

9.2 Synchronous functions

The following sequence diagram shows the function `Fls_GetJobResult` as an example for the synchronous functions of this module. The same sequence applies also to the functions `Fls_GetStatus` and `Fls_SetMode`.

Figure 0.54
«module»
NvM
«module»
MemIf
«module»
Fee
«module»
Fls
MemIf_GetJobResult(MemIf_JobResultType, uint8)
Fee_GetJobResult(MemIf_JobResultType)
Fls_GetJobResult(MemIf_JobResultType)
Fls_GetJobResult
()
Fee_GetJobResult
()
MemIf_GetJobResult
()

Figure 2: Fls_GetJobResult

52 of 74
Document ID 25: AUTOSAR_SWS_FlashDriver
- AUTOSAR confidential -

3.2 01_Retrieval Augmented Generation System

3.2.1 Overview

In the dynamic landscape of conversational AI, the integration of advanced techniques is crucial to enhance the capabilities of Chatbots. One such groundbreaking approach is Retrieval Augmented Generation (RAG), which combines the strengths of information retrieval and natural language generation. This technique not only improves the quality of responses but also contributes significantly to the overall performance of Chatbots.

3.2.1.1 What is Retrieval Augmented Generation (RAG)?

Retrieval Augmented Generation (RAG) is an innovative approach that seamlessly blends information retrieval and language generation to create more contextually relevant and coherent responses. Unlike traditional Chatbot architectures, RAG leverages a dual-stage process where it first retrieves relevant information from a knowledge base and then generates a response based on the retrieved context.

3.2.1.2 Key Features

1. **Contextual Relevance**: RAG ensures that the generated responses are contextually relevant by incorporating information retrieved from a knowledge base. This leads to more accurate and meaningful interactions.
2. **Dynamic Adaptability**: The technique allows Chatbots to dynamically adapt to different user queries and contexts, providing a more personalized and user-friendly experience.
3. **Enhanced Understanding**: By utilizing information retrieval, RAG improves the Chatbot's understanding of user inputs, leading to more coherent and context-aware responses.

3.2.1.3 Why RAG in Building a Chatbot?

3.2.1.3.1 Improved Response Quality

RAG significantly enhances the quality of responses by drawing upon a vast knowledge base. This ensures that the Chatbot's replies are well-informed and contextually appropriate, leading to a more satisfying user experience.

3.2.1.3.2 Contextual Awareness

In traditional Chatbot architectures, maintaining contextual awareness can be challenging. RAG addresses this issue by dynamically adapting to user inputs and maintaining a continuous understanding of the conversation context, resulting in more natural and engaging interactions.

3.2.1.3.3 Personalization

Users expect Chatbots to understand their unique needs and preferences. RAG facilitates personalization by retrieving information specific to individual users, creating a more tailored and user-centric conversational experience.

3.2.1.3.4 Handling Diverse Queries

As users present diverse queries, the ability of a Chatbot to handle a wide range of topics becomes crucial. RAG's integration of information retrieval allows Chatbots to provide accurate and relevant responses across various domains.

3.2.1.4 Pre-tasks for Applying RAG

Before implementing the Retrieval Augmented Generation (RAG) technique, it's essential to perform the following pre-tasks:

3.2.1.4.1 Create Middle Data

3.2.1.4.1.1 Extraction from Raw Documents

To kickstart the RAG system, the extraction of data from raw documents (in formats like Word or PDF) is required. This extraction process aims to convert the information into an intermediate JSON format. This middle data serves as the foundation for subsequent processing stages.

Please refer to 02_ Create The Middle Database - ASD-VN-MCAL-CICD-RAC - Confluence ASR [Live] (renesas.eu) (see page 51)

3.2.1.4.2 Create Final Data

3.2.1.4.2.1 Data Transformation and Post-processing

The final data, directly consumed by the RAG system, is derived from the middle data. This task involves the transformation of the extracted information into a format suitable for RAG. Additionally, it includes the segmentation of data items into smaller, more manageable chunks and the implementation of post-processing techniques to enhance the quality of information.

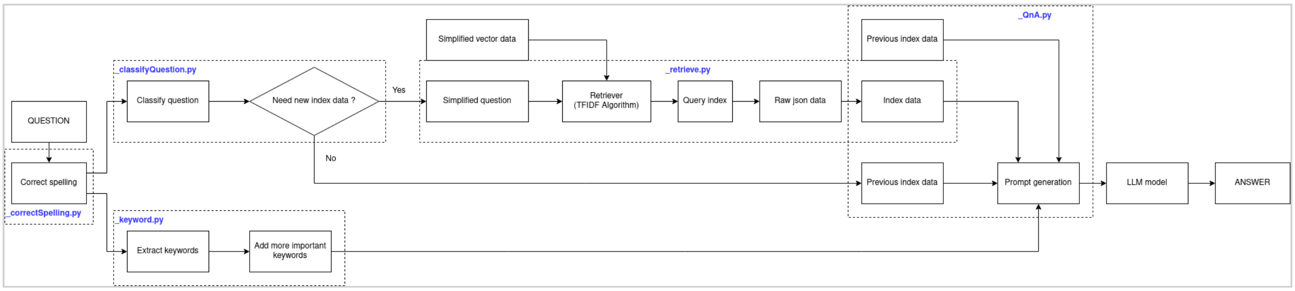
These pre-tasks are integral steps in preparing the groundwork for the successful implementation of Retrieval Augmented Generation in your Chatbot development process.

Please refer to 03_Create The Final Database - ASD-VN-MCAL-CICD-RAC - Confluence ASR [Live] (renesas.eu) (see page 57)

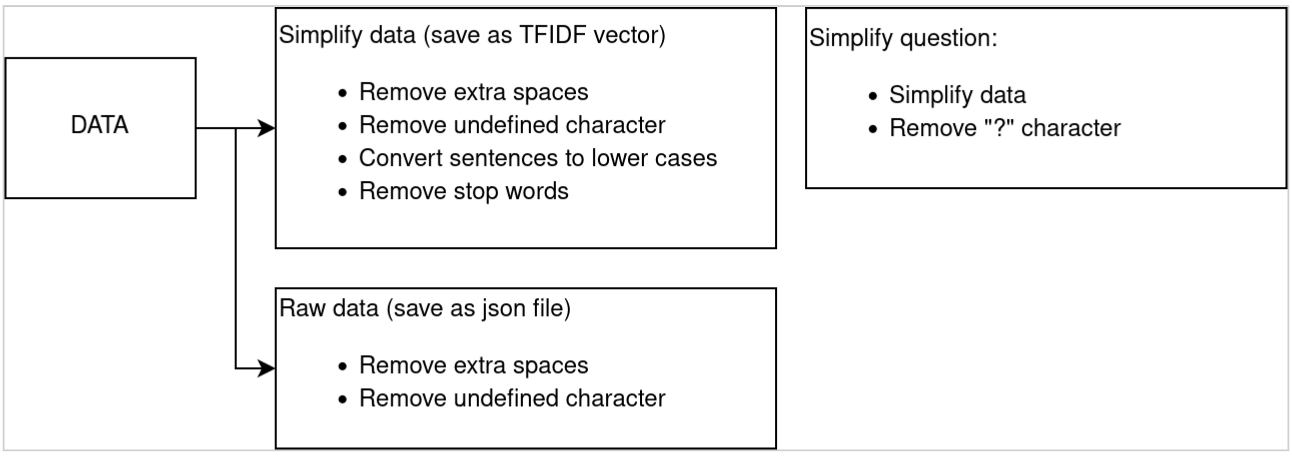
3.2.2 RAG Approaches

3.2.2.1 Stub Solution

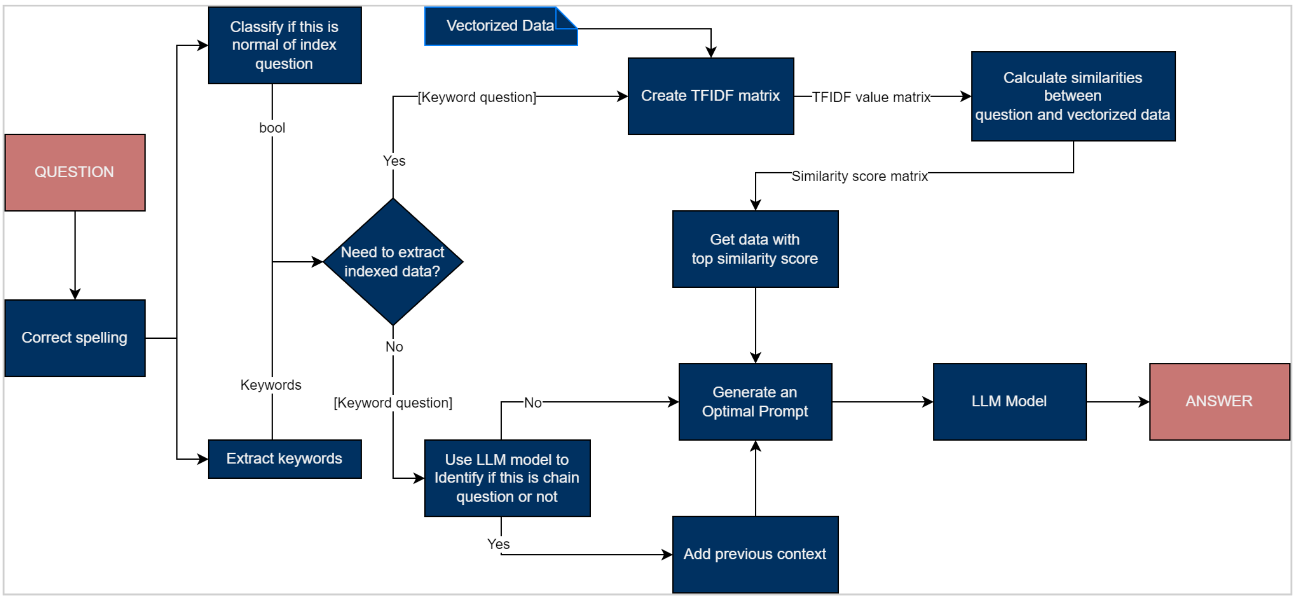
3.2.2.1.1 Overall



3.2.2.1.2 Data simplification


DATA
Simplify data (save as TFIDF vector)
• Remove extra spaces
• Remove undefined character
• Convert sentences to lower cases
• Remove stop words
Raw data (save as json file)
• Remove extra spaces
• Remove undefined character
Simplify question:
• Simplify data
• Remove "?" character

3.2.2.2 Upgraded Solution (in-used)


QUESTION
Correct spelling
Classify if this is normal of index question
bool
Extract keywords
Keywords
Need to extract indexed data?
Yes
No
[Keyword question]
[Keyword question]
Vectorized Data
Create TFIDF matrix
TFIDF value matrix
Calculate similarities between question and vectorized data
Similarity score matrix
Get data with top similarity score
Use LLM model to Identify if this is chain question or not
No
Yes
Add previous context
Generate an Optimal Prompt
LLM Model
ANSWER

3.2.3 Details

3.2.3.1 Retrieving Data with TF-IDF Values

3.2.3.1.1 Compute TF-IDF Values

In the context of information retrieval, the TF-IDF (Term Frequency-Inverse Document Frequency) algorithm is employed to assess the significance of terms in a document concerning a corpus. Here's a more detailed breakdown of TF-IDF:

- **Term Frequency (TF)**: This component measures how frequently a term appears in a document. It is calculated as the number of occurrences of a term in a document divided by the total number of terms in that document. A higher TF value indicates a higher importance of the term in the document.

$$\mathrm{TF}(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

- **Inverse Document Frequency (IDF)**: IDF evaluates the importance of a term across the entire corpus. It is calculated as the logarithm of the total number of documents divided by the number of documents containing the term. A higher IDF value signifies that the term is less common across the corpus.

$$\mathrm{IDF}(t, D) = \log\left(\frac{\text{Total number of documents in corpus } D}{\text{Number of documents containing term } t \text{ in corpus } D}\right)$$

- **TF-IDF Score**: The TF-IDF score is obtained by multiplying the TF and IDF values for each term. This score reflects the importance of a term in a specific document relative to its importance in the entire corpus.

$$\text{TF-IDF}(t, d, D) = \mathrm{TF}(t, d) \times \mathrm{IDF}(t, D)$$

3.2.3.1.2 Query Processing

- When a user submits a query, preprocess the query text in the same way you preprocessed the document text.
- Calculate the TF-IDF score for each term in the query.
- Compare the TF-IDF scores of the query terms with the TF-IDF scores of the terms in your documents.

3.2.3.1.3 Calculate the Similarity

- Compute the cosine similarity between the TF-IDF vector of the query and the TF-IDF vectors of your documents. Cosine similarity ranges from -1 to 1, with 1 indicating identical documents.
- Rank the documents based on their similarity scores.
- Return the top-ranked documents as the semantic matches for the user's query.

3.2.3.1.4 Implementation

DEMO IMPLEMENTATION	
	```
	from sklearn.feature_extraction.text import
	 TfidfVectorizer
	from sklearn.metrics.pairwise import cosine_similarity

	# Example data
	documents = ["JSON document 1 text", "JSON document 2
	text", ...]
	user_query = "User's query text"

	# TF-IDF Vectorization
	vectorizer = TfidfVectorizer()
	tfidf_matrix = vectorizer.fit_transform(documents +
	 [user_query])

	# Cosine Similarity Calculation
	cosine_similarities = cosine_similarity(tfidf_matrix[-1]
	, tfidf_matrix[:-1])

	# Get top semantic matches
	top_matches = [(index, score) for index, score in
	enumerate(cosine_similarities[0])]
	top_matches.sort(key=lambda x: x[1], reverse=True)

	# Retrieve top 5 matches (for example)
	top_5_matches = top_matches[:5]
	```

IN SOURCE CODE	
	```
	from sklearn.metrics.pairwise import cosine_similarity
	# Load vectorizer from pickle file
	with open(vectorizer_path, 'rb') as file:
	    vectorizer = pickle.load(file)
	# Load vectorized_texts from pickle file
	with open(vectorized_texts_path, 'rb') as file:
	    vectorized_texts = pickle.load(file)
	# TFIDF Algorithm Runv
	## Pre-process and compute TF-IDF value for the
	question
	vectorized_keywords =
	 vectorizer.transform([simplifyQuestion(question)])
	## Get cosine similarity score vector of vectorized
	texts and vectorized question
	```

DEMO IMPLEMENTATION

```python
from sklearn.feature_extraction.text import
 TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# Example data
documents = ["JSON document 1 text", "JSON document 2
text", ...]
user_query = "User's query text"

# TF-IDF Vectorization
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(documents +
 [user_query])

# Cosine Similarity Calculation
cosine_similarities = cosine_similarity(tfidf_matrix[-1]
, tfidf_matrix[:-1])

# Get top semantic matches
top_matches = [(index, score) for index, score in
enumerate(cosine_similarities[0])]
top_matches.sort(key=lambda x: x[1], reverse=True)

# Retrieve top 5 matches (for example)
top_5_matches = top_matches[:5]
```

```python
similarity_scores =
 cosine_similarity(vectorized_keywords,
vectorized_texts)[0]
# Filter out the scores
match_indices = list(range(len(similarity_scores)))
## Thresholding the scores
filtered_similarity_scores = [score for score in
 similarity_scores if score > self.threshold]
## Thresholding the indices
filtered_match_indices = [index for index, score in zip(
match_indices, similarity_scores) if score > self.thres
hold]
```

3.2.3.2 Other Approach for Retrieving Data

This approach is potentially better than the implemented one. The next developers could try to integrate this new solution.

3.2.3.2.1 General

1. **Choose an NLP Library or API:** Select an NLP library or API that provides semantic matching capabilities. Popular choices include:
    - **spaCy**: A Python library for NLP.
    - **NLTK**: Natural Language Toolkit for Python.
    - **BERT (Bidirectional Encoder Representations from Transformers)**: A pre-trained deep learning model for natural language understanding.
2. **Preprocess the Data:**
    - Load your JSON data into memory.
    - Preprocess the text data to clean and tokenize it. Tokenization involves breaking down sentences into individual words or tokens.
3. **Embedding the Data:**
    - Convert your text data into numerical vectors. This can be done using pre-trained word embeddings (e.g., Word2Vec, GloVe) or language models like BERT.
    - If using BERT, you can use the BERT tokenizer and model to convert your text into contextual embeddings.
4. **User Query Processing:**
    - Preprocess the user's query in a similar way to how you preprocessed your data.
    - Convert the user's query into numerical vectors using the same method used for the data.
5. **Semantic Matching:**
    - Compute the similarity between the user's query vector and each vector in your database.
    - Common methods include cosine similarity or dot product between vectors.
    - Libraries like spaCy have built-in similarity functions.
6. **Ranking and Retrieval:**
    - Rank the data based on the similarity scores.
    - Retrieve the top N documents that match the user's query.
7. **Response Generation:**
    - Present the retrieved data to the user in a meaningful way.

3.2.3.2.2 Implementation

```python
import spacy

# Load the spaCy model
nlp = spacy.load("en_core_web_md")
```

```
# Assuming 'data' is a list of dictionaries with a 'text' key containing your text
data
data = [...]

# Preprocess and embed data
embedded_data = [nlp(item['text']).vector for item in data]

def get_top_semantic_match(user_query, embedded_data, top_n=5):
    # Preprocess user query
    user_query_vector = nlp(user_query).vector

    # Calculate similarity scores
    similarity_scores = [cosine_similarity(user_query_vector, doc_vector) for
 doc_vector in embedded_data]

    # Get indices of top N documents
    top_indices = sorted(range(len(similarity_scores)), key=lambda i:
similarity_scores[i], reverse=True)[:top_n]

    # Retrieve top N documents
    top_documents = [data[i] for i in top_indices]

    return top_documents

# Example usage
user_query = "How does our company handle customer complaints?"
top_matches = get_top_semantic_match(user_query, embedded_data)
print(top_matches)
```

3.2.3.3 Final Prompt

```
prompt_for_common_query = '''### Instruction: AI Document Assistant
You are an AI Document assistant. User will you give you a task. Your goal is to
complete the task as faithfully as you can.
Task: Response politely to the following user's query or question:
History: {history}
Information: {input}
Question: {question}
### Response:
'''

prompt_for_specific_inquiry = '''### Instruction: AI Document Assistant
You are a helpful AI Document assistant and you can understand any long document.
Your goal is to obtain the following pieces of information to get your understand.
Then answer the question at the end of the information pieces.
If you need more information in the previous context, please refer to the History
part. Now read the following information pieces and answer the question:
History: {history}
Information: {input}
```

```
Question: {question}
### Response:
'''
```

There will be replacements in the prompt:

- {history}: replaced with the previous conversation if exists.
- {input}: replaced with top retrieved data items.
- {question}: replaced with user's query.

3.2.4 Basic Usage

The code can be used locally because it does not load the LLM model. The main purpose of the code is to retrieve data and generate the final prompt

3.2.4.1 Requirement

Suppose that you have already set up the environment for development. If not yet, please refer to:

- 01_Set Up Environment Instruction - ASD-VN-MCAL-CICD-RAC - Confluence ASR [Live] (renesas.eu) (see page 25)
- 02_Deployment Instruction (see page 32)

3.2.4.2 Explain config parameters

Modify **./mcal-guru/03_Prompt/_config.yaml** referred to the below explanation.

```
environment: 'local'                                  # choose between 'local' and 'colab'
root_local: 'D:/Training/AI_for_MCAL/source_code/Knowledge_Transfer/MCAL_BOT_Resources/'
root_colab: '/content/drive/MyDrive/Chatbot/'

need_decode: True
key_path: 'final_data/encryption_key.key'            # key use in encrypt data
data: 'final_data/v4_encrypted/'                     # path to data folder (used to retrieve data)
embedding_model: 'model/all-mpnet-base-v2-table'     # path to embedding model (used to extract keywords)
LLM_model: 'model/gpt4-x-vicuna-13B-GPTQ'            # path to LLM model (load tokenizer only) used to split data when create final data and
                                                      # reduce index data to make token_length < max_token in retrieve data
max_token: 1500                                       # max_token used to reduce index data
retriever: 'TFIDF'                                    # TFIDF only
threshold: 0.0                                        # threshold to remove irrelevant index data
```

```
number_of_query: 20                                   # number of index data being retrieved
link_number: 3                                        # number of preference link
debug: True                                           # print log or not
```

3.2.4.3 Generate final prompt from a question

```
from _QnA import QnA

prompt_for_common_query = '''### Instruction: AI Document Assistant
You are an AI Document assistant. User will you give you a task. Your goal is to complete the task as faithfully as you can.
Task: Response politely to the following user's query or question:
History: {history}
Information: {input}
Question: {question}
### Response:
'''

prompt_for_specific_inquiry = '''### Instruction: AI Document Assistant
You are a helpful AI Document assistant and you can understand any long
 document. Your goal is to obtain the following pieces of information to get your understand. Then answer the question at the end of the information pieces.
If you need more information in the previous context, please refer to the History part. Now read the following information pieces and answer the question:
History: {history}
Information: {input}
Question: {question}
### Response:
'''

relate_question_prompt = """Generate at least 2 questions that related to below conversation.
{history}
### Instruction: {question}
### Response: {answer}
### Related question:
"""
qna = QnA()

questions = {"hello guy": False,
"How are you?": False,
"Compare AD vs UD": False,
"Hi, it is very interesting, could you say more about it?": True,
"What is Driver Functional Design ?": False,
"What is ADC_init?": False,
}
```

```python

data = {'token': '2804bad6fe94a55f18b2b37e300919a5fd517b95aa81e95db574c0ba069a3740', 'question': 'what could you do?', 'path': 'internal/'}
dict_items = list(questions.items())
index = 1
for question, type in dict_items[1:]:
    previous_question, previous_type = dict_items[index-1]

    if "history" in data: history = data["history"]
    else: history = []

    if "path" in data: path = data["path"]
    else: path = ""

    print("\n\n####################################################################################\n")
    final_prompt, references, wkproduct_links, guideline_links, guidelines, conversation_type = qna.generate_prompt(path, question, prompt_for_common_query, prompt_for_specific_inquiry, history)

    completed_final_prompt = final_prompt
    if conversation_type == True:
        additional_field_index = final_prompt.find(f"### Instruction:")
        if additional_field_index != -1:
            completed_final_prompt = final_prompt[:additional_field_index] + previous_question + "\n" + final_prompt[additional_field_index:]

    print(f"Question: {question}")
    print(f"Conversation Chain?: Infer:{conversation_type} - Ground Truth: {type}")
    index += 1

    print("############## FINAL PROMPT ##############")
    print(completed_final_prompt)
```

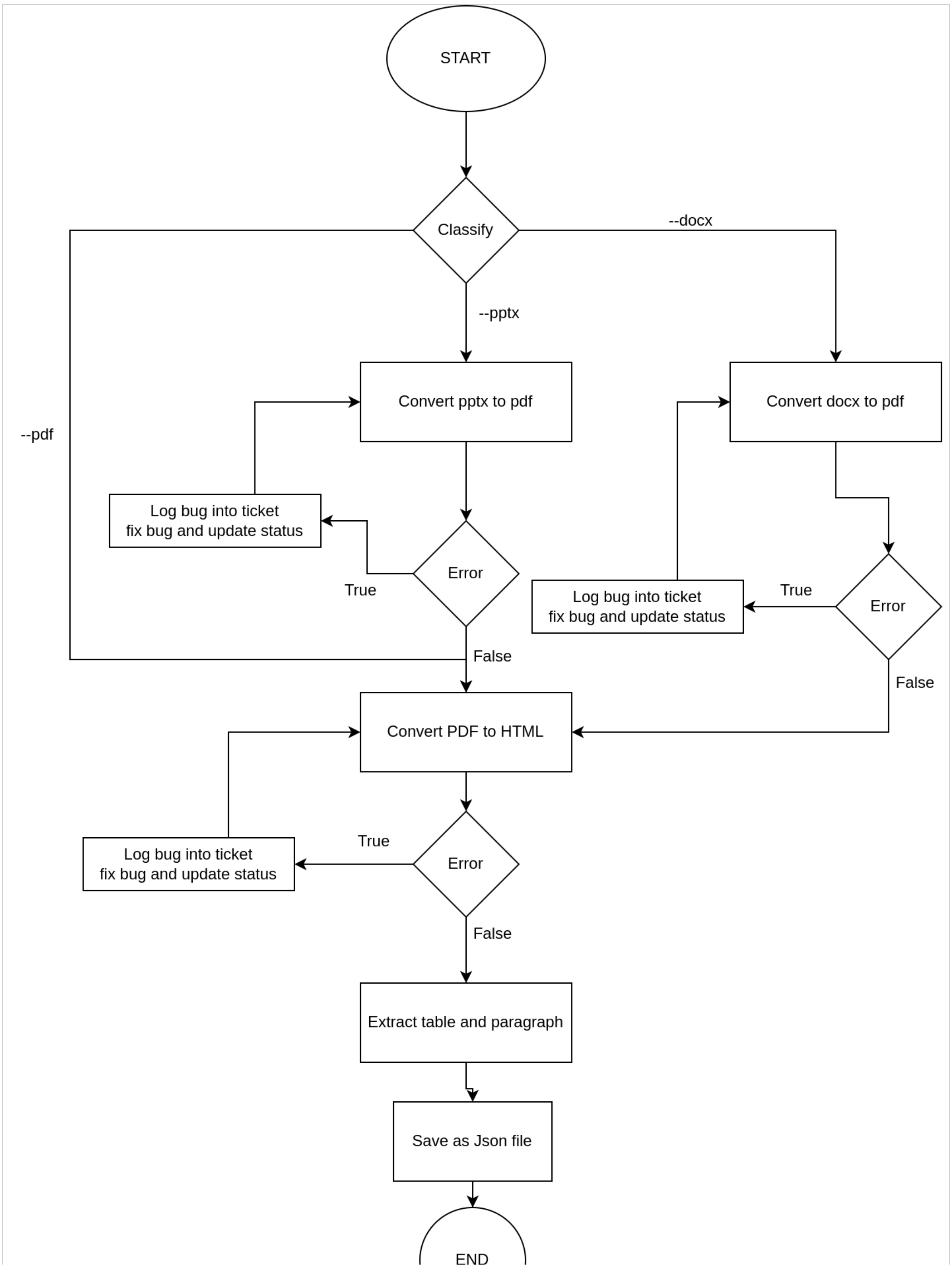
3.3 02_ Create The Middle Database

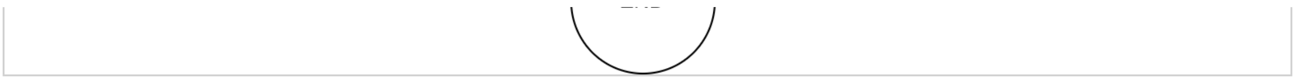
3.3.1 General

- Creating the middle database is the task of converting documents in many formats (.docx, .pdf, .pptx, .html....) to JSON format.
- In sprint 1, we created
    - AUTOSAR Database:
        - R.4.3.1
        - R19-11

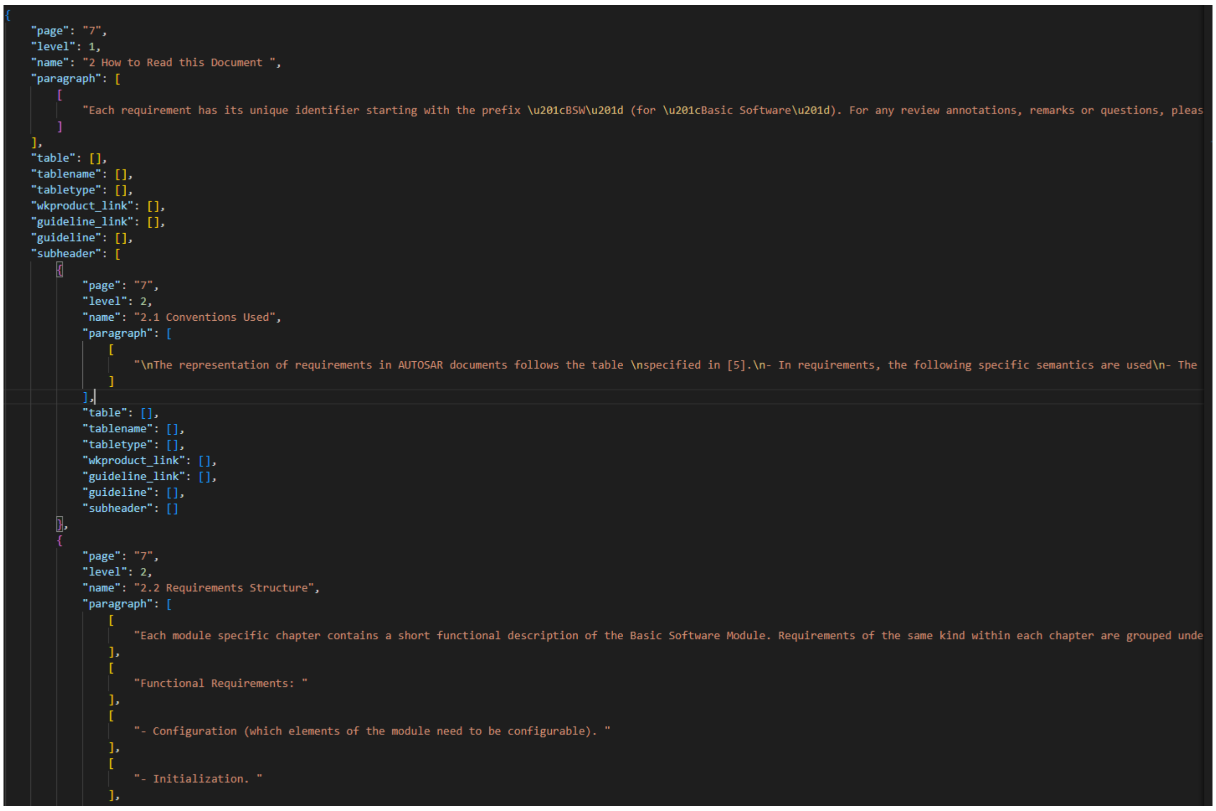
            - R20-11
            - R21-11
        - Hardware Manual Database
            - RH850
                - Core
                - D1x
                - E2x
                - U2x
            - RCAR
                - ArmCore
                - R-CarGen4
- Remaining:
    - Hardware Manual
        - RCAR
            - ArmCore
                - ARMv8
                    - A64_v83A_ISA_xml_00bet5
                    - Address_TranslationARMv8-R
                    - ARMv83A-SysReg-00bet5
                    - CCI-500
                    - Cortex-A53
                    - Cortex-A57
                    - Cortex-A76
                    - Cortex-R52
                    - DynamicIQDynamIQ Shared Unit (DSU)
                    - GIC
                    - Presentation_slides
                    - SCMI
                    - SMMU

3.3.2 Convert flow


START
Classify
--docx
--pptx
--pdf
Convert pptx to pdf
Convert docx to pdf
Log bug into ticket
fix bug and update status
Error
True
False
Log bug into ticket
fix bug and update status
True
Error
False
Convert PDF to HTML
True
Log bug into ticket
fix bug and update status
Error
False
Extract table and paragraph
Save as Json file
END



3.3.3 JSON format



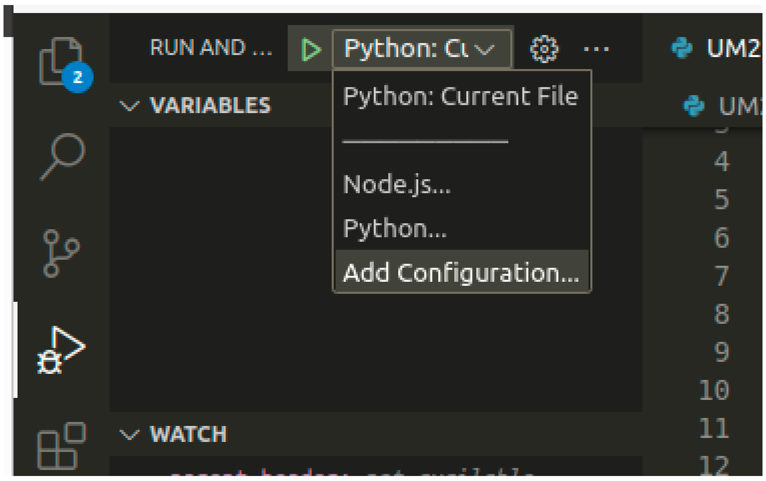
```
{
    "page": "7",
    "level": 1,
    "name": "2 How to Read this Document ",
    "paragraph": [
        [
            "Each requirement has its unique identifier starting with the prefix \u201cBSW\u201d (for \u201cBasic Software\u201d). For any review annotations, remarks or questions, pleas
        ]
    ],
    "table": [],
    "tablename": [],
    "tabletype": [],
    "wkproduct_link": [],
    "guideline_link": [],
    "guideline": [],
    "subheader": [
        {
            "page": "7",
            "level": 2,
            "name": "2.1 Conventions Used",
            "paragraph": [
                [
                    "\nThe representation of requirements in AUTOSAR documents follows the table \nspecified in [5].\n- In requirements, the following specific semantics are used\n- The
                ]
            ],
            "table": [],
            "tablename": [],
            "tabletype": [],
            "wkproduct_link": [],
            "guideline_link": [],
            "guideline": [],
            "subheader": []
        },
        {
            "page": "7",
            "level": 2,
            "name": "2.2 Requirements Structure",
            "paragraph": [
                [
                    "Each module specific chapter contains a short functional description of the Basic Software Module. Requirements of the same kind within each chapter are grouped unde
                ],
                [
                    "Functional Requirements: "
                ],
                [
                    "- Configuration (which elements of the module need to be configurable). "
                ],
                [
                    "- Initialization. "
                ],
```

3.3.4 Converting Script

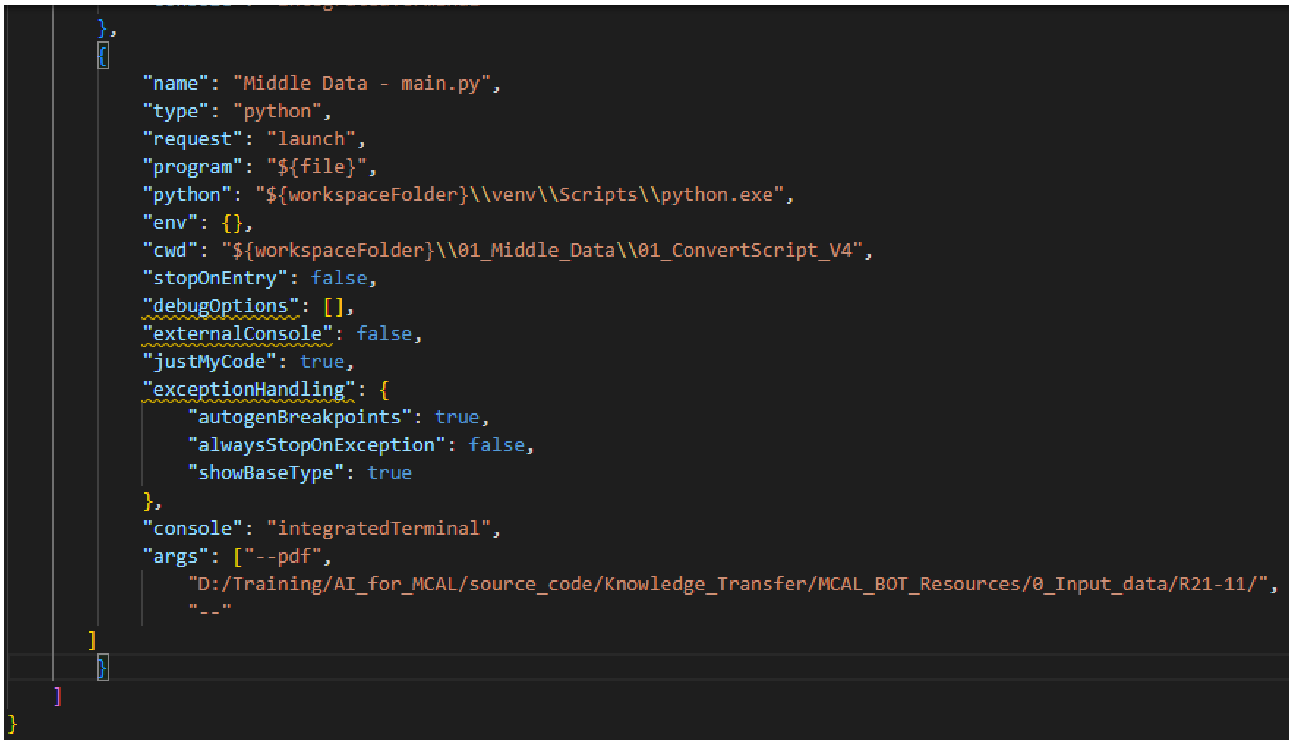
3.3.4.1 To run Autosar2middledatabase.py or UM2middledatabase.py, there are 2 methods:

3.3.4.1.1 Run in debug mode :

- Create a launch.json file



- Add configuration.



```
        },
        {
            "name": "Middle Data - main.py",
            "type": "python",
            "request": "launch",
            "program": "${file}",
            "python": "${workspaceFolder}\\venv\\Scripts\\python.exe",
            "env": {},
            "cwd": "${workspaceFolder}\\01_Middle_Data\\01_ConvertScript_V4",
            "stopOnEntry": false,
            "debugOptions": [],
            "externalConsole": false,
            "justMyCode": true,
            "exceptionHandling": {
                "autogenBreakpoints": true,
                "alwaysStopOnException": false,
                "showBaseType": true
            },
            "console": "integratedTerminal",
            "args": ["--pdf",
                "D:/Training/AI_for_MCAL/source_code/Knowledge_Transfer/MCAL_BOT_Resources/0_Input_data/R21-11/",
                "--"
        ]
        }
    ]
}
```

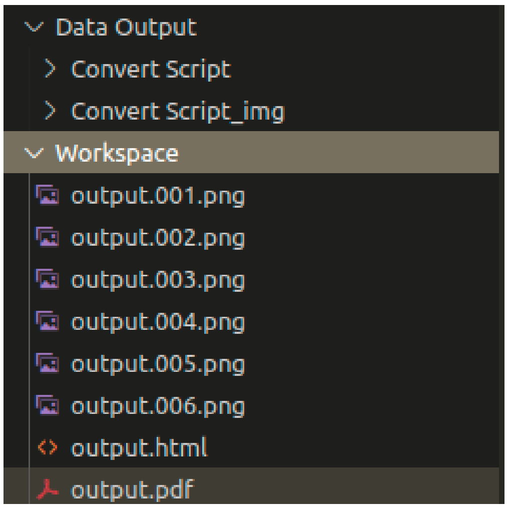
- Argument Description:
  - Argument 1: There are 3 options --pdf for pdf file, --pptx for PowerPoint file, and --docx for Word file).
  - Argument 2: Path to the folder you want to convert.
  - Argument 3: There are 2 options: --update in this mode when converting script only convert the file which does not exist in the 'Data Output' folder, '–' for convert all.

3.3.4.1.2 Run in terminal:

- python3 Autosar2middledatabase.py --pdf/ or pptx/ or docx --path_to_data --update/or not

3.3.4.1.3 Outputs:

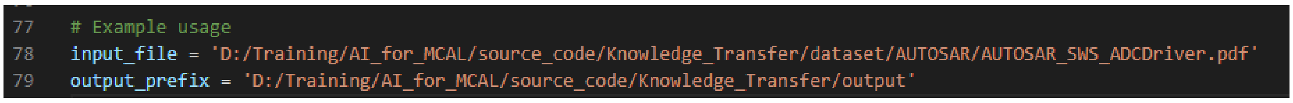
- A Workspace folder to contain files while converting
- A Data Output folder to contain output files (JSON and text)



```
Data Output
  Convert Script
  Convert Script_img
Workspace
  output.001.png
  output.002.png
  output.003.png
  output.004.png
  output.005.png
  output.006.png
  output.html
  output.pdf
```

3.3.4.2 To run SplitPDF.py

- Firstly, modify these lines with your data path:



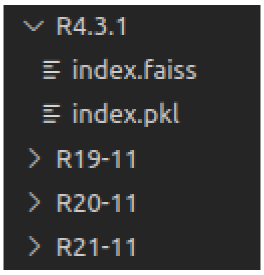
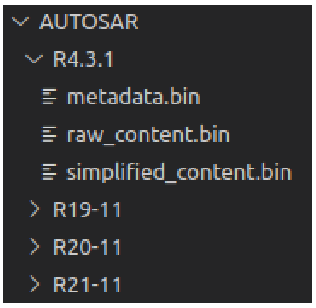
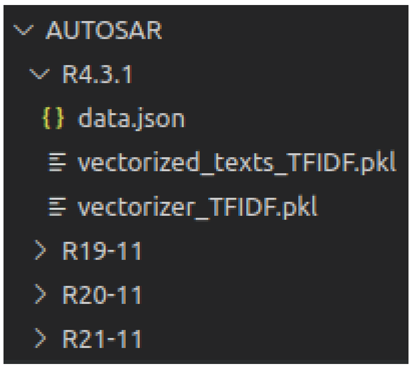
```python
# Example usage
input_file = 'D:/Training/AI_for_MCAL/source_code/Knowledge_Transfer/dataset/AUTOSAR/AUTOSAR_SWS_ADCDriver.pdf'
output_prefix = 'D:/Training/AI_for_MCAL/source_code/Knowledge_Transfer/output'
```

- Run script: python SplitPDF.py

3.4 03_Create The Final Database

3.4.1 Data Structure

3.4.1.1 Saving Formats

Format 1	**Format 2**	**Format 3**
Use vector store FAISS to create	Save the array as a .bin file	Save raw data as JSON file and simplified data as TFIDF vector (.pkl file)
 R4.3.1 / index.faiss / index.pkl / R19-11 / R20-11 / R21-11	 AUTOSAR / R4.3.1 / metadata.bin / raw_content.bin / simplified_content.bin / R19-11 / R20-11 / R21-11	 AUTOSAR / R4.3.1 / data.json / vectorized_texts_TFIDF.pkl / vectorizer_TFIDF.pkl / R19-11 / R20-11 / R21-11 **=> At this time this is the best format with the main advantage of not taking time and memory to load and store variables**

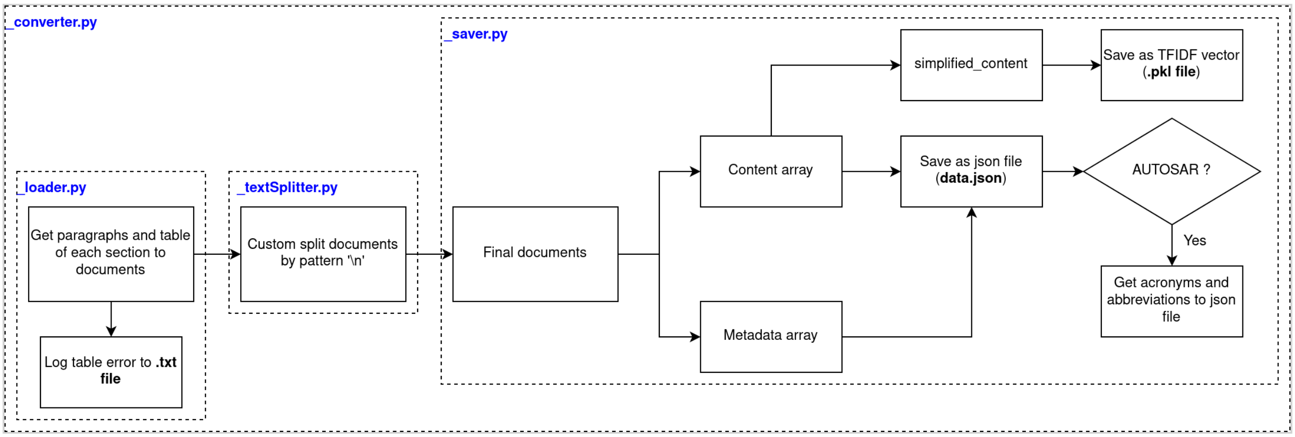
3.4.1.2 Final Data Versions

Version 1	**Version 2**	**Version 3**	**Version 4**
• **Middle data: V2** • **Save as file save format 1** • Vectorstores: FAISS • Embedding type: HuggingFaceEmbeddings • Embedding model: deepset/all-mpnet-base-v2-table • Splitter: CharacterTextSplitter • separator = "\n", • chunk_size = 2000, • chunk_overlap = 0, • length_function = len,	• **Middle data: V3** • **Save as file save format 2** • Split into 2 data files: + Simplified data: remove unnecessary information for better retrieval. + Raw data: Necessary information data are given to prompt the model to answer. • Splitter: CharacterTextSplitter • separator = ".\n", • chunk_size = 500, • chunk_overlap = 0, • length_function = len,	• **Middle data: V3** • **Save as file save format 3** • Split into 2 data files: + Simplified data: save as TFIDF vector (TFIDF is a retrieve algorithm) + Raw data: save as JSON file • Splitter: CharacterTextSplitter • separator = ".\n", • chunk_size = 500, • chunk_overlap = 0, • length_function = len,	• **Middle data: V4** • **Save as file save format 3** • Split into 2 data files: + Simplified data: save as TFIDF vector (TFIDF is a retrieve algorithm) + Raw data: save as JSON file

Version 1	**Version 2**	**Version 3**	**Version 4**
• Update from v3: - Fix some errors in arranged data - Remove '[' and ']' character - Fix vertical table - **Remove all documents that have the same content** • Arrange data: - Paragraph - Table 1: Heading 1 | Heading 2 | Heading 3 A | B | C D | E | F - Table 2: Heading 4 | A Heading 5 | B Heading 6 | C • Replace \n with '' in the horizontal table • Replace '[' and ']' with '' in paragraph => A document contains data of "a paragraph" or "a row of the table", including: Paragraph **or** {Heading 1: A, Heading 2: B, Heading 3: C} **or** {Heading 1: D, Heading 2: E, Heading 3: F} **or** Heading 4: A - Heading 5: B - Heading 6: C **or** Paragraph ...	• Update from V4: - Remove tables of content - Add the requirement name to the corresponding table - Connect 2 tables separated by pages - Split into 2 data: + Raw data: remove extra spaces and remove undefined characters. + Simplified data: remove extra spaces, remove undefined characters, convert to lowercase, and remove stop words • Arrange data: The same as version 4, but combine all paragraphs in one page into a chunk	• Arrange data: The same as version 2 • Encrypted version: each content is encrypted to improve security.	• Arrange data: - Split document by header - For each header content, split into texts and tables + Texts: separate texts into small paragraphs, each paragraph in one document + Tables: each table in one document - Map requirement text with relevant table • Encrypted version: each content is encrypted to improve security.

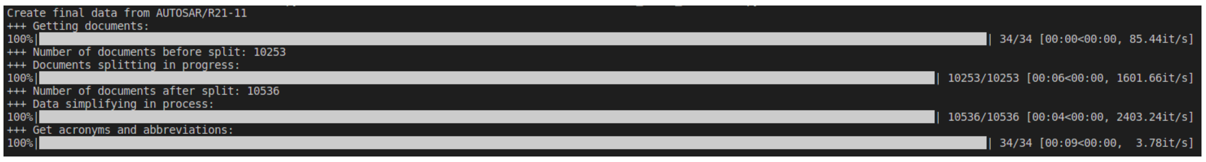
Version 1	**Version 2**	**Version 3**	**Version 4**
• **Only a paragraph or a row of tables in one chunks** • **Retrieve the exact number of documents < number of tokens for a question**	• **All paragraphs in one page or a row of tables in one chunk** • **Retrieve the exact number of documents < number of tokens for a question. Combine algorithm and embedding model to retrieve.**	• **All paragraphs in one page or a row of tables in one chunk** • **Retrieve the exact number of documents < number of tokens for a question. Use only the TFIDF algorithm to retrieve**	• **Retrieve the exact number of documents < number of tokens for a question. Use only the TFIDF algorithm to retrieve** **=> At this time this is the best version**

3.4.2 Convert Flow


_converter.py
_saver.py
_loader.py
Get paragraphs and table of each section to documents
Log table error to .txt file
_textSplitter.py
Custom split documents by pattern '\n'
Final documents
Content array
Metadata array
simplified_content
Save as TFIDF vector (.pkl file)
Save as json file (data.json)
AUTOSAR ?
Yes
Get acronyms and abbreviations to json file

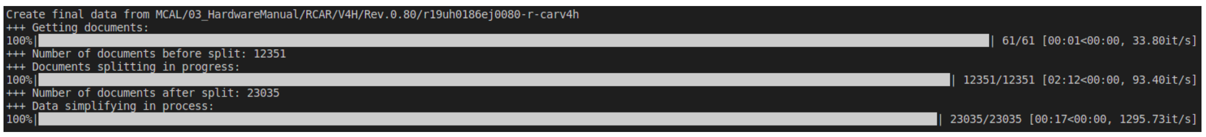
Example Terminal log:

- AUTOSAR



```
Create final data from AUTOSAR/R21-11
+++ Getting documents:
100%|████████████████████████████████████████████████████████████| 34/34 [00:00<00:00, 85.44it/s]
+++ Number of documents before split: 10253
+++ Documents splitting in progress:
100%|█████████████████████████████████████████████████████████| 10253/10253 [00:06<00:00, 1601.66it/s]
+++ Number of documents after split: 10536
+++ Data simplifying in process:
100%|█████████████████████████████████████████████████████████| 10536/10536 [00:04<00:00, 2403.24it/s]
+++ Get acronyms and abbreviations:
100%|████████████████████████████████████████████████████████████| 34/34 [00:09<00:00,  3.78it/s]
```

- Hardware Manual



```
Create final data from MCAL/03_HardwareManual/RCAR/V4H/Rev.0.80/r19uh0186ej0080-r-carv4h
+++ Getting documents:
100%|████████████████████████████████████████████████████████████| 61/61 [00:01<00:00, 33.80it/s]
+++ Number of documents before split: 12351
+++ Documents splitting in progress:
100%|██████████████████████████████████████████████████████████| 12351/12351 [02:12<00:00, 93.40it/s]
+++ Number of documents after split: 23035
+++ Data simplifying in process:
100%|█████████████████████████████████████████████████████████| 23035/23035 [00:17<00:00, 1295.73it/s]
```

3.4.3 Configuration

Edit in config.yaml

```yaml
debug: True
# Print log while running
middle_version_path: 'D:/Training/AI_for_MCAL/source_code/
Knowledge_Transfer/MCAL_BOT_Resources/02_Final_data/middle_data/tmps'
     # Path to middle data
key_path: 'D:/Training/AI_for_MCAL/source_code/Knowledge_Transfer/
MCAL_BOT_Resources/02_Final_data/middle_data/encryption_key.key'       #
Path to key -> used for encode documents
tokenizer_path: 'D:/Training/AI_for_MCAL/source_code/Knowledge_Transfer/
MCAL_BOT_Resources/model/OpenOrca-Platypus2-13B-GPTQ'            # Path to
tokenizer of LLM model -> used to split documents
embedding_dir: 'D:/Training/AI_for_MCAL/source_code/Knowledge_Transfer/
MCAL_BOT_Resources/model/all-mpnet-base-v2-table'                 # Path
to embedding model -> used to save database in save_format 1
```

3.4.4 Basic Usage

The output of the running will contain an internal and an external folder

- **Internal**: used to store the database of AUTOSAR and MCAL. In the internal folder, the file structure is the same as the middle, and input data.
- **External**: used to store the database of an external file chat feature.

There are 2 modes to generate final data: encrypt and not encrypt.

- If encrypted final data is generated, configure **./mcal-guru/03_Prompt/config.yaml**
    - need_decode: True
- If final data is generated without encryption, configure **./mcal-guru/03_Prompt/config.yaml**
    - need_decode: False

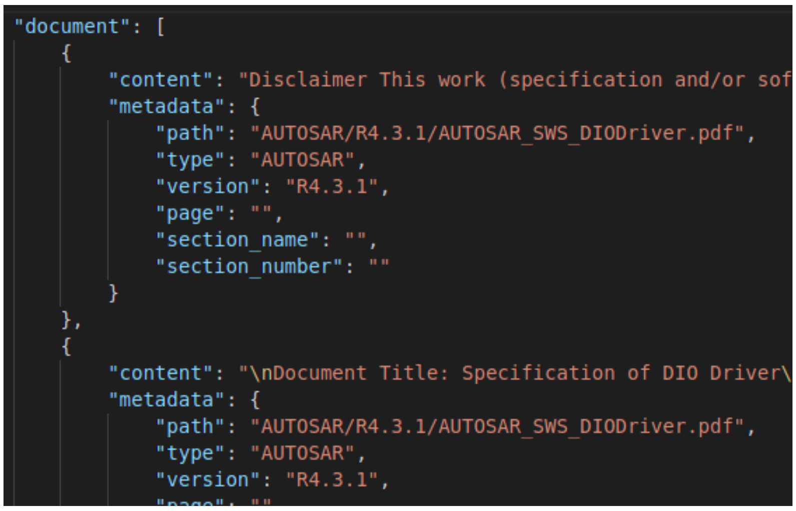
3.4.4.1 Create Final Data without Encryption

- Modify **./mcal-guru/02_Final_Data/ConvertScript/main.py** as below:

```python
from _converter import CONVERTER
from extern_variables import *

# Convert AUTOSAR document
convert = CONVERTER(type = 'AUTOSAR')
convert.run(module_sub_path = 'AUTOSAR',
            output_path = OUTPUT_PATH,
            save_format = 3, encode = False)
```

```

# Convert Hardware Manual document
convert = CONVERTER(type = 'BASE')
convert.run(module_sub_path = 'HardwareManual',
            output_path = OUTPUT_PATH,
            save_format = 3, encode = False)
```

- Sample Output:



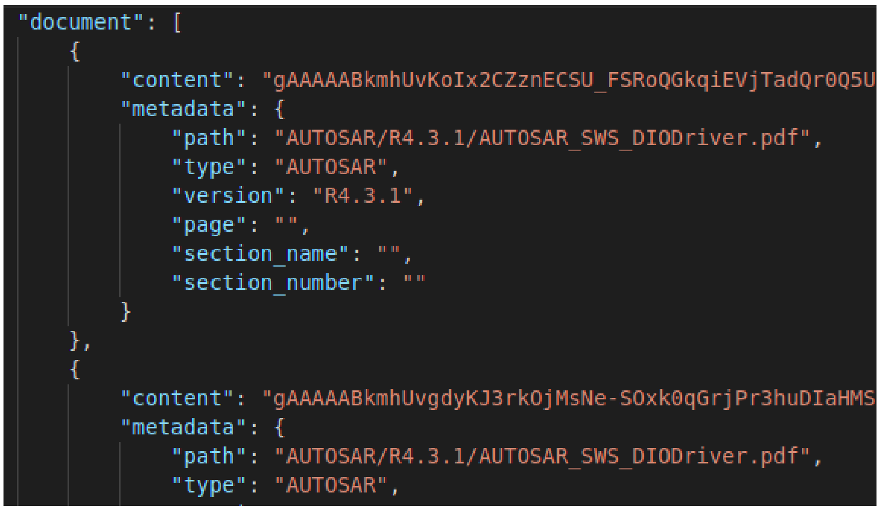
```
"document": [
    {
        "content": "Disclaimer This work (specification and/or sof
        "metadata": {
            "path": "AUTOSAR/R4.3.1/AUTOSAR_SWS_DIODriver.pdf",
            "type": "AUTOSAR",
            "version": "R4.3.1",
            "page": "",
            "section_name": "",
            "section_number": ""
        }
    },
    {
        "content": "\nDocument Title: Specification of DIO Driver\
        "metadata": {
            "path": "AUTOSAR/R4.3.1/AUTOSAR_SWS_DIODriver.pdf",
            "type": "AUTOSAR",
            "version": "R4.3.1",
```

3.4.4.2 Create Final Data with Encryption

- Modify **./mcal-guru/02_Final_Data/ConvertScript/main.py** as below:

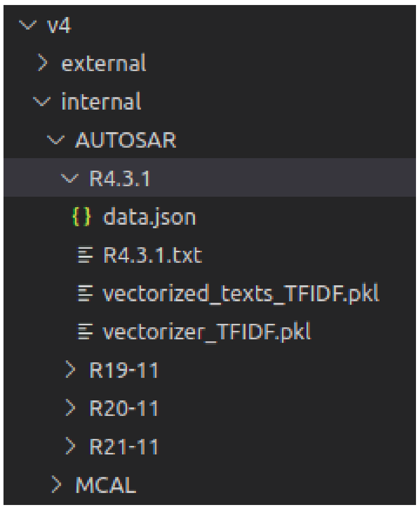
```
from _converter import CONVERTER
from extern_variables import *

# Convert AUTOSAR document
convert = CONVERTER(type = 'AUTOSAR')
convert.run(module_sub_path = 'AUTOSAR',
            output_path = OUTPUT_PATH,
            save_format = 3, encode = True)

# Convert Hardware Manual document
convert = CONVERTER(type = 'BASE')
convert.run(module_sub_path = 'HardwareManual',
            output_path = OUTPUT_PATH,
            save_format = 3, encode = True)
```

- Sample Output:



```
"document": [
    {
        "content": "gAAAAABkmhUvKoIx2CZznECSU_FSRoQGkqiEVjTadQr0Q5U
        "metadata": {
            "path": "AUTOSAR/R4.3.1/AUTOSAR_SWS_DIODriver.pdf",
            "type": "AUTOSAR",
            "version": "R4.3.1",
            "page": "",
            "section_name": "",
            "section_number": ""
        }
    },
    {
        "content": "gAAAAABkmhUvgdyKJ3rkOjMsNe-SOxk0qGrjPr3huDIaHMS
        "metadata": {
            "path": "AUTOSAR/R4.3.1/AUTOSAR_SWS_DIODriver.pdf",
            "type": "AUTOSAR",
```

3.4.5 Output



```
v v4
  > external
  v internal
    v AUTOSAR
      v R4.3.1
        {} data.json
        R4.3.1.txt
        vectorized_texts_TFIDF.pkl
        vectorizer_TFIDF.pkl
      > R19-11
      > R20-11
      > R21-11
    > MCAL
```

- data.json: Final data main content.
- vectorized_texts_TFIDF.pkl and vectorizer_TFIDF.pkl: vector data of TFIDF algorithm → used to get query index → use query index to get index data from data.json
- .txt file: table log error: this error is logged when I can not extract table information → usually diagram to the table in the document

3.5 04_[AI-Chatbot] Finetune LLM Model

3.5.1 1. General

The main technique used in this topic is QLoRA Fine-Tuning.

QLoRA, short for Quantization Low-rank adaptation, is a powerful technique designed to enhance the efficiency and adaptability of pre-trained language models during the fine-tuning process. Originating from the realms of quantization and low-rank matrix factorization, QLoRA addresses the challenges associated with deploying large language models to specific tasks or domains, where computational resources and model size constraints are critical considerations.

The purpose of doing so is to minimize the hardware needed to finetune, and thus be able to finetune the model on Google Colab.

3.5.2 Adapting QLoRA for LLM Fine-Tuning

Main libraries:

- **bitandbytes:** library to quantize the model to 4-bit.
- **Peft:** library to do the LoRA Fine-Tuning.

Reference document:

- Youtube instruction: https://www.youtube.com/watch?v=NRVaRXDoI3g&t=631s
- Colab instruction: https://colab.research.google.com/drive/1Vvju5kOyBsDr7RX_YAvp6ZsSOoSMjhKD?usp=sharing
- HuggingFace instruction: https://huggingface.co/blog/4bit-transformers-bitsandbytes
- QLoRA: https://arxiv.org/pdf/2305.14314.pdf
- LoRA: https://arxiv.org/pdf/2106.09685.pdf | https://huggingface.co/docs/peft/conceptual_guides/lora

3.5.3 2. Source Code (Google Colab notebook): https://colab.research.google.com/drive/1StdWgvrZ9wNPvkNtE8hzLc08zVgZcHl4?usp=sharing

3.5.4 3. Prepare Training Data

The type of training data is **DatasetDict** - a type from the **datasets** library.

To create **DatasetDict**, first need a JSON data in the format like:

Json data format

```json
{
    "document": [
        {
            "content": "content 1",
            "metadata": {
                "source": "source 1"
            }
        },
        {
            "content": "content 2",
            "metadata": {
                "source": "source 2"
            }
        }
    ]
}
```

I just use the **content** field to finetune, so we might not need the **metadata** field.

Here is an example code to load JSON file and create **DatasetDict**, this code is included in my Google Collab:

```python
from datasets import Dataset, DatasetDict
import json
with open('/content/drive/MyDrive/Chatbot/finetune/AUTOSAR/final_data/data_v2.json', 'r') as file:
    data = json.load(file)
dataset = Dataset.from_dict(data)
dataset = dataset.map(lambda example: {"content": example["document"]["content"]}, remove_columns=["document"]) #only use content field
data = DatasetDict({"train": dataset})
data = data.map(lambda samples: tokenizer(samples["content"]), batched=True) # use content to finetune => need to tokenize the content
```

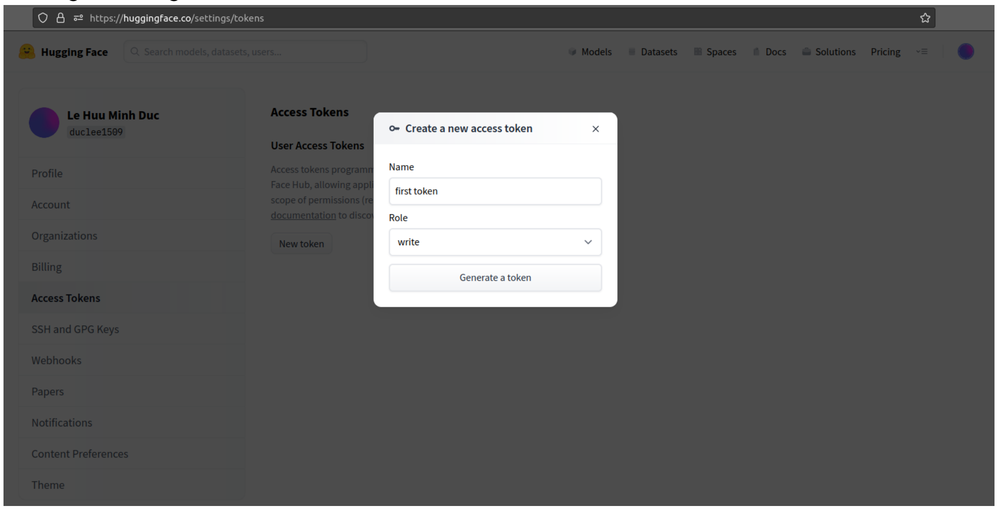
I have tried 3 types of data:

- **data_v1**: collect data from 7 links about AUTOSAR/MCAL general knowledge, then split them into many paragraphs manually | code: **data_v1.py**
- **data_v2**: collect 7 paragraphs from 7 links about AUTOSAR/MCAL, then give them to chatGPT and ask to generate instruction - response pairs | code: **data_v2.py**
- **data_v3**: content from final data of 2 documents: **AUTOSAR_SWS_ADCDriver** and **AUTOSAR_SWS_CANDriver** | code: **data_v3.py**

3.5.4.1 4. Finetuning

To finetune, run the following blocks in Google Colab notebook: (I have commented in each code block): Init → Basic model → Data → Train → Push model to the **huggingface hub**.

1. Init
    - Login to huggingface_hub: copy and paste the Access token in the personal hugging face setting to Google Colab.



Create a new access token

Name: first token

Role: write

2. Basic model: use the hugging face model type, not the GPTQ model type.
3. Data
4. Train
    a. LoRA Parameters config:

```python
from peft import LoraConfig, get_peft_model

config = LoraConfig(
    r=16,
    lora_alpha=16,
    # target_modules=["query_key_value"],
    lora_dropout=0.01,
    bias="none",
    task_type="CAUSAL_LM"
)
```

Terminology	**Explanation**
r	• Increasing r would make the model more expressive by allowing it to capture more diverse patterns and relationships. However, it would also lead to increased computational complexity.

Terminology	Explanation
lora_alpha	• Increasing lora_alpha would emphasize the influence of global information, potentially enabling the model to capture broader context and long-range dependencies. However, setting it too high might make the model less attentive to local patterns
lora_dropout	• Increasing `lora_dropout` would enhance regularization, making the model more robust and less prone to overfitting. However, setting it too high might negatively affect the model's ability to learn and generalize from the data.

b. Hyper-parameters config:

```python
import transformers

tokenizer.pad_token = tokenizer.eos_token

trainer = transformers.Trainer(
    model=model,
    train_dataset=data["train"],
    args=transformers.TrainingArguments(
        per_device_train_batch_size=1,
        gradient_accumulation_steps=4,
        warmup_steps=100,
        max_steps=500,
        learning_rate=2e-4,
        fp16=True,
        logging_steps=1,
        output_dir="outputs",
        optim="paged_adamw_8bit"
    ),
    data_collator=transformers.DataCollatorForLanguageModeling(tokenizer, mlm=False),
)
model.config.use_cache = False  # silence the warnings. Please re-enable for inference!
trainer.train()
```

5. Push the model to hugging face

a. Change model name
b. You must log in to huggingface_hub in the init step to be able to push the model hugging face
c. The model pushed to hugging face is not the entire model, it's just an adapt weight used to add on the basic model

3.5.4.2 5. Inference

1. Used to evaluate the results before and after finetuning
2. Run the following blocks in Google Colab.
   a. **Before finetune**: Init → Basic model → Text generation code → Prompt → Before finetune
   b. **After finetune**: Init → Finetuned model → Text generation code → Prompt → After finetune

3.6 05_[AI-UT] Fine-Tune LLM Model

3.6.1 General

Using the same technique as fine-tuning the LLM model for Chatbot, which is called QLoRA.

3.6.2 Coding Model

- As of now, I've identified a suitable LLM model for the coding mission, known as **deepseek-ai/deepseek-coder-6.7b-instruct.**
- This model has been obtained from deepseek-ai/deepseek-coder-6.7b-instruct at main (huggingface.co)[2]
- DeepSeek Coder is composed of a series of code language models, each trained from scratch on 2T tokens. For coding capabilities, DeepSeek Coder achieves state-of-the-art performance among open-source code models on multiple programming languages and various benchmarks.
- You might find more information at:
  - deepseek-ai/DeepSeek-Coder: DeepSeek Coder: Let the Code Write Itself (github.com)[3]
  - [ASDMCALCICD-459] [AI - UT] Leverage Coding LLM Model & Coding Embedding Model - JIRA ASR-d [Live] (renesas.eu)[4]

3.6.3 Training Data Preparation

I've come up with the training data in this format:

```
{
```

2 https://huggingface.co/deepseek-ai/deepseek-coder-6.7b-instruct/tree/main
3 https://github.com/deepseek-ai/deepseek-coder
4 https://jira.renesas.eu/browse/ASDMCALCICD-459

```
    "Input": <function code>,
    "Output":
    {
        "Input"
         {
             "Input_Param_001: {"Type:" <param type>, "Range": <param range>, "Name":
<name of param>, "Value": <param value>}
             ....
        }
        "Output"
         {
             "Output_Param_001: {"Type:" <param type>, "Range": <param range>, "Name"
: <name of param>, "Value": <param value>}
             ....
        }
    }
}
```

Here is an example:

3.6.4 Choosing An Appropriate Max Length

- With the prepared data, we analyze and find the suitable max length.

- Choose the suitable config length = 1024.

3.6.5 Fine-Tuning Process

3.6.5.1 LoRA configuration

```
from peft import LoraConfig, get_peft_model

config = LoraConfig(
    r=32,
    lora_alpha=64,
    target_modules=[
        "q_proj",
        "k_proj",
        "v_proj",
        "o_proj",
        "gate_proj",
        "up_proj",
        "down_proj",
        "lm_head",
    ],
    bias="none",
    lora_dropout=0.05,  # Conventional
    task_type="CAUSAL_LM",
)

model = get_peft_model(model, config)
print_trainable_parameters(model)

# trainable params: 85041152 || all params: 3837112320 || trainable%:
2.2162799758751914
```

3.6.5.2 Hyper-Parameters

3.6.5.3 Traning Loss

3.6.6 Testing

- INPUT:

```
Input = """
### Input Function:
** Function Name        : Adc_AdcDeInit
**
** Service ID           : NA
**
** Description          : This internal function performs the de-initialization of
the device-specific ADC hardware
**                        registers.
**
** Sync/Async           : Synchronous
**
** Re-entrancy          : Non-Reentrant
**
** Input Parameters     : LucHwUnitIndex
**
** InOut Parameters     : None
**
```

```
** Output Parameters    : None
**
** Return Parameter     : None
**
** Preconditions        : None
**
** Global Variables     : Adc_GpHwUnitConfig
**
** Functions invoked    : Adc_HwTriggerDeInit, Adc_TrackHoldDeInit,
Adc_DeInitWaitTime
**
** Registers Used       : ADCXnDFASENTSGER, ADCXnSDVCLMINTER, ADCXnSGDIAGPCCR0,
ADCXnSGDIAGPCCR1, ADCXnSGDIAGVCR0,
**                        ADCXnSGDIAGVCR1, ADCXnSGDIAGVCR2, ADCXnSTPDCR0,
ADCXnSTPDCR1, ADCXnSTPDCR2, ADCXnSGDIAGCR
**
** Reference ID         : ADC_DUD_ACT_085, ADC_DUD_ACT_085_REG001,
ADC_DUD_ACT_085_REG002,
** Reference ID         : ADC_DUD_ACT_085_REG003, ADC_DUD_ACT_085_REG004,
ADC_DUD_ACT_085_REG005,
** Reference ID         : ADC_DUD_ACT_085_REG006, ADC_DUD_ACT_085_REG007,
ADC_DUD_ACT_085_REG008,
** Reference ID         : ADC_DUD_ACT_085_REG009, ADC_DUD_ACT_085_REG010,
ADC_DUD_ACT_085_REG011
*************************************************************************************
**********************************/
#if (ADC_DEINIT_API == STD_ON)

#define ADC_START_SEC_PRIVATE_CODE
#include
"Adc_Mapping.h"
                          /* PRQA S 5087 # JV-01 */

FUNC(void, ADC_PRIVATE_CODE) Adc_AdcDeInit(const uint8 LucHwUnitIndex)
/* PRQA S 1532 # JV-01 */
{
  P2CONST(Adc_HwUnitConfigType, AUTOMATIC, ADC_CONFIG_DATA) LpHwUnitConfig;
  P2VAR(volatile Adc_ConfigRegisters, AUTOMATIC, REGSPACE) LpAdcRegisters;

  /* Initialize the local variable for HW unit configuration */
  LpHwUnitConfig = &Adc_GpHwUnitConfig[LucHwUnitIndex];
/* PRQA S 2824 # JV-01 */
  /* Read the user base configuration address of the HW unit */
  LpAdcRegisters = LpHwUnitConfig->pHwUnitBaseAddress;
/* PRQA S 2814 # JV-01 */

  /* Initialize the device-specific ADC hardware registers to their default values */
  LpAdcRegisters->usADCXnDFASENTSGER = ADC_WORD_ZERO;
/* PRQA S 2814 # JV-01 */
  LpAdcRegisters->ucADCXnSDVCLMINTER = ADC_BYTE_ZERO;
  LpAdcRegisters->ulADCXnSGDIAGPCCR0 = ADC_DWORD_ZERO;
  LpAdcRegisters->ulADCXnSGDIAGPCCR1 = ADC_DWORD_ZERO;
  LpAdcRegisters->ucADCXnSGDIAGVCR0  = ADC_BYTE_ZERO;
```

```
    LpAdcRegisters->ucADCXnSGDIAGVCR1  = ADC_BYTE_ZERO;
    LpAdcRegisters->ucADCXnSGDIAGVCR2  = ADC_BYTE_ZERO;
    LpAdcRegisters->ucADCXnSTPDCR0     = ADC_BYTE_ZERO;
    LpAdcRegisters->ucADCXnSTPDCR1     = ADC_BYTE_ZERO;
    LpAdcRegisters->ucADCXnSTPDCR2     = ADC_BYTE_ZERO;
    LpAdcRegisters->ulADCXnSGDIAGCR    = ADC_DWORD_ZERO;

    #if (ADC_ENABLE_ADC_TSEL == STD_ON)
    /* Check if HW unit has ADC trigger selection register */
    if (NULL_PTR != LpHwUnitConfig->pSgTriggReg)
    {
      /* De-initialization of ADC trigger selection register */
      Adc_HwTriggerDeInit(LucHwUnitIndex);
    } /* else: No action required */
    #endif /* (ADC_ENABLE_ADC_TSEL == STD_ON) */

    #if (ADC_TRACK_AND_HOLD == STD_ON)
    /* Check if HW unit has Track and Hold groups */
    if ((uint8)0U != LpHwUnitConfig->ucTrackHoldEnable)
    {
      /* De-initialization of Track and Hold */
      Adc_TrackHoldDeInit(LucHwUnitIndex);
    } /* else: No action required */
    #endif

    /* Check if HW unit has wait time configuration */
    if (ADC_WAITTIME_DISABLED != LpHwUnitConfig->ucWaitTimeIndex)
    {
      /* De-initialization of wait time configuration */
      Adc_DeInitWaitTime(LucHwUnitIndex);
    } /* else: No action required */
  }

  ### Output Test Specification:
  """
```

- OUTPUT:

3.6.7 Conclusion

- I observed a gradual decrease in the training loss.
- However, the test results did not meet expectations.
- When I input the 'Adc_Deinit' function, the output was chaotic. The words were jumbled and did not align with the training data, making it challenging to find a coherent meaning.
- This issue might be attributed to the training data or the nature of the model itself, as it wasn't primarily trained for coding purposes.

I intend to explore alternative solutions.

4 04_Autonomous_Agent_Design

4.1 [MCAL_AI] Agents Implementation

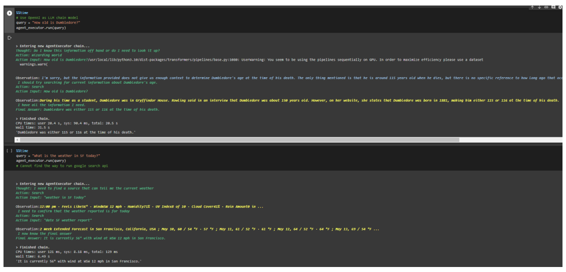
4.1.1 Introduction

This article is the detailed introduction about some agents and its results.

4.1.2 Implementation Results

4.1.2.1 Action Agent Executor

Action Agents are conventional and good for small tasks. It makes decisions on the actions to take and executes them step by step.

Flowchart	Result
Action_Agent.drawio.pdf	

```
%%time
# Use OpenAI as LLM chain model
query = "How old is Dumbledore?"
agent_executor.run(query)

> Entering new AgentExecutor chain...
Thought: Do I know this information off hand or do I need to look it up?
Action: Wizarding World
Action Input: How old is Dumbledore?/usr/local/lib/python3.10/dist-packages/transformers/pipelines/base.py:1080: UserWarning: You seem to be using the pipelines sequentially on GPU. In order to maximize efficiency please use a dataset
  warnings.warn(


Observation: I'm sorry, but the information provided does not give us enough context to determine Dumbledore's age at the time of his death. The only thing mentioned is that he is around 115 years old when he dies, but there is no specific reference to how long ago that occ
 I should try searching for current information about Dumbledore's age.
Action: Search
Action Input: How old is Dumbledore?

Observation:During his time as a student, Dumbledore was in Gryffindor House. Rowling said in an interview that Dumbledore was about 150 years old. However, on her website, she states that Dumbledore was born in 1881, making him either 115 or 116 at the time of his death.
 I have all the information I need.
Final Answer: Dumbledore was either 115 or 116 at the time of his death.

> Finished chain.
CPU times: user 20.4 s, sys: 90.4 ms, total: 20.5 s
Wall time: 31.5 s
'Dumbledore was either 115 or 116 at the time of his death.'
```

```
%%time
query = "What is the weather in SF today?"
agent_executor.run(query)
# Cannot find the way to run google search api

> Entering new AgentExecutor chain...
Thought: I need to find a source that can tell me the current weather
Action: Search
Action Input: "weather in SF today"

Observation:12:00 pm · Feels Like56° · WindWSW 12 mph · Humidity71% · UV Index8 of 10 · Cloud Cover41% · Rain Amount0 in ...
 I need to confirm that the weather reported is for today
Action: Search
Action Input: "date SF weather report"

Observation:2 Week Extended Forecast in San Francisco, California, USA ; May 10, 60 / 54 °F · 57 °F ; May 11, 61 / 52 °F · 61 °F ; May 12, 64 / 52 °F · 64 °F ; May 13, 69 / 54 °F ...
 I now know the final answer
Final Answer: It is currently 56° with wind at WSW 12 mph in San Francisco.

> Finished chain.
CPU times: user 121 ms, sys: 8.18 ms, total: 129 ms
Wall time: 6.49 s
'It is currently 56° with wind at WSW 12 mph in San Francisco.'
```

4.1.2.2 Plan-and-Execute Agent

For more complex or long-running tasks, the initial planning step helps to maintain long-term objectives and focus. However, that comes at the expense of generally more calls and higher latency

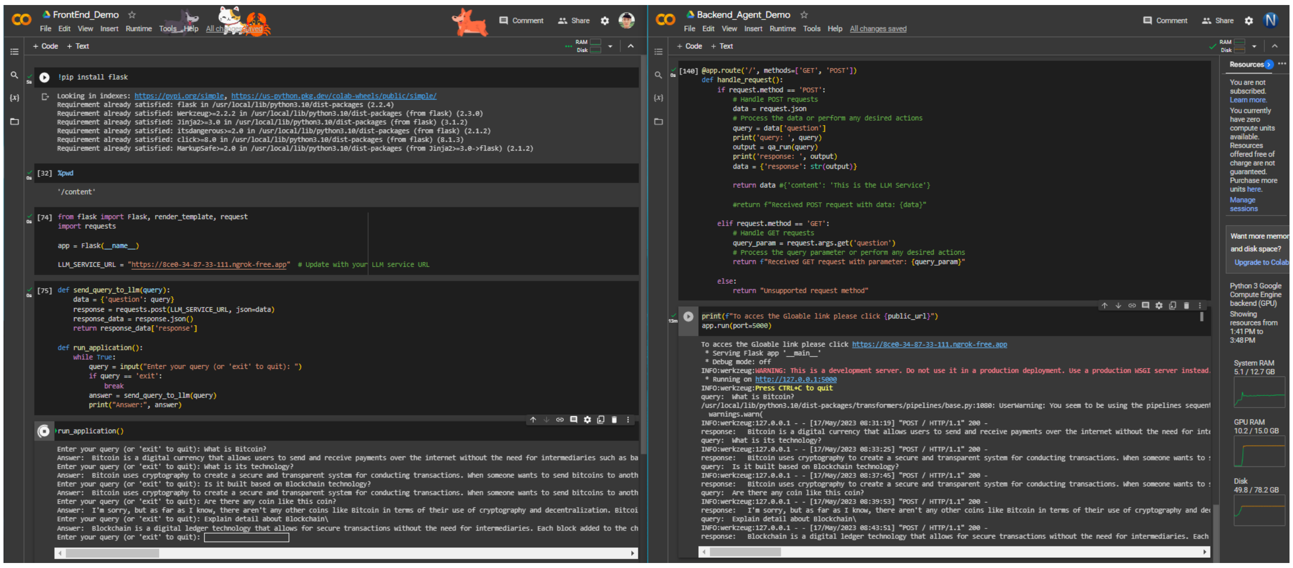
Flowchart	Result
Plan-and-Execute Agents.drawio.pdf	refer to this link for result: BabyAGI with LangChain Agent (Google) - Colaboratory[5]

4.1.2.3 Backend Agent Development

This task involves developing a backend service for an Autonomous Agent. The agent will be based on the LLM model and will generate answers based on the given query. The backend service should provide an **API key/URL** for other services to call.

Below is my result.

5 https://colab.research.google.com/drive/1IzYc6kLzuRMXfUI4aCNxWCYfzZHtdCf-#scrollTo=f7957b51



```
!pip install flask

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: flask in /usr/local/lib/python3.10/dist-packages (2.2.4)
Requirement already satisfied: Werkzeug>=2.2.2 in /usr/local/lib/python3.10/dist-packages (from flask) (2.3.0)
Requirement already satisfied: Jinja2>=3.0 in /usr/local/lib/python3.10/dist-packages (from flask) (3.1.2)
Requirement already satisfied: itsdangerous>=2.0 in /usr/local/lib/python3.10/dist-packages (from flask) (2.1.2)
Requirement already satisfied: click>=8.0 in /usr/local/lib/python3.10/dist-packages (from flask) (8.1.3)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from Jinja2>=3.0->flask) (2.1.2)

[32] %pwd

'/content'

[74] from flask import Flask, render_template, request
     import requests

     app = Flask(__name__)

     LLM_SERVICE_URL = "https://8ce0-34-87-33-111.ngrok-free.app"  # Update with your LLM service URL

[75] def send_query_to_llm(query):
         data = {'question': query}
         response = requests.post(LLM_SERVICE_URL, json=data)
         response_data = response.json()
         return response_data['response']

     def run_application():
         while True:
             query = input("Enter your query (or 'exit' to quit): ")
             if query == 'exit':
                 break
             answer = send_query_to_llm(query)
             print("Answer:", answer)

run_application()

Enter your query (or 'exit' to quit): What is Bitcoin?
Answer:  Bitcoin is a digital currency that allows users to send and receive payments over the internet without the need for intermediaries such as ba
Enter your query (or 'exit' to quit): What is its technology?
Answer:  Bitcoin uses cryptography to create a secure and transparent system for conducting transactions. When someone wants to send bitcoins to anoth
Enter your query (or 'exit' to quit): Is it built based on Blockchain technology?
Answer:  Bitcoin uses cryptography to create a secure and transparent system for conducting transactions. When someone wants to send bitcoins to anoth
Enter your query (or 'exit' to quit): Are there any coin like this coin?
Answer:  I'm sorry, but as far as I know, there aren't any other coins like Bitcoin in terms of their use of cryptography and decentralization. Bitcoi
Enter your query (or 'exit' to quit): Explain detail about Blockchain\
Answer:  Blockchain is a digital ledger technology that allows for secure transactions without the need for intermediaries. Each block added to the ch
Enter your query (or 'exit' to quit):
```

```
[140] @app.route('/', methods=['GET', 'POST'])
      def handle_request():
          if request.method == 'POST':
              # Handle POST requests
              data = request.json
              # Process the data or perform any desired actions
              query = data['question']
              print('query: ', query)
              output = qa_run(query)
              print('response: ', output)
              data = {'response': str(output)}

              return data #{'content': 'This is the LLM Service'}

              #return f"Received POST request with data: {data}"

          elif request.method == 'GET':
              # Handle GET requests
              query_param = request.args.get('question')
              # Process the query parameter or perform any desired actions
              return f"Received GET request with parameter: {query_param}"

          else:
              return "Unsupported request method"

print(f"To acces the Gloable link please click {public_url}")
app.run(port=5000)

To acces the Gloable link please click https://8ce0-34-87-33-111.ngrok-free.app
 * Serving Flask app '__main__'
 * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
query:  What is Bitcoin?
/usr/local/lib/python3.10/dist-packages/transformers/pipelines/base.py:1080: UserWarning: You seem to be using the pipelines sequent
  warnings.warn(
INFO:werkzeug:127.0.0.1 - - [17/May/2023 08:31:19] "POST / HTTP/1.1" 200 -
response:   Bitcoin is a digital currency that allows users to send and receive payments over the internet without the need for inte
query:  What is its technology?
INFO:werkzeug:127.0.0.1 - - [17/May/2023 08:33:25] "POST / HTTP/1.1" 200 -
response:   Bitcoin uses cryptography to create a secure and transparent system for conducting transactions. When someone wants to s
query:  Is it built based on Blockchain technology?
INFO:werkzeug:127.0.0.1 - - [17/May/2023 08:37:45] "POST / HTTP/1.1" 200 -
response:   Bitcoin uses cryptography to create a secure and transparent system for conducting transactions. When someone wants to s
query:  Are there any coin like this coin?
INFO:werkzeug:127.0.0.1 - - [17/May/2023 08:39:53] "POST / HTTP/1.1" 200 -
response:   I'm sorry, but as far as I know, there aren't any other coins like Bitcoin in terms of their use of cryptography and de
query:  Explain detail about Blockchain\
INFO:werkzeug:127.0.0.1 - - [17/May/2023 08:43:51] "POST / HTTP/1.1" 200 -
response:   Blockchain is a digital ledger technology that allows for secure transactions without the need for intermediaries. Each
```

4.1.3 Conclusion

The original Action Agent and the Plan-and-Execute Agent are deeply based on the inference ability of the LLM model.

=> We can not control the task with specific and detailed steps.

I am going to define another method with can handle our pre-defined tasks.

4.2 [MCAL_AI] LangChain Concepts Investigation

4.2.1 Introduction

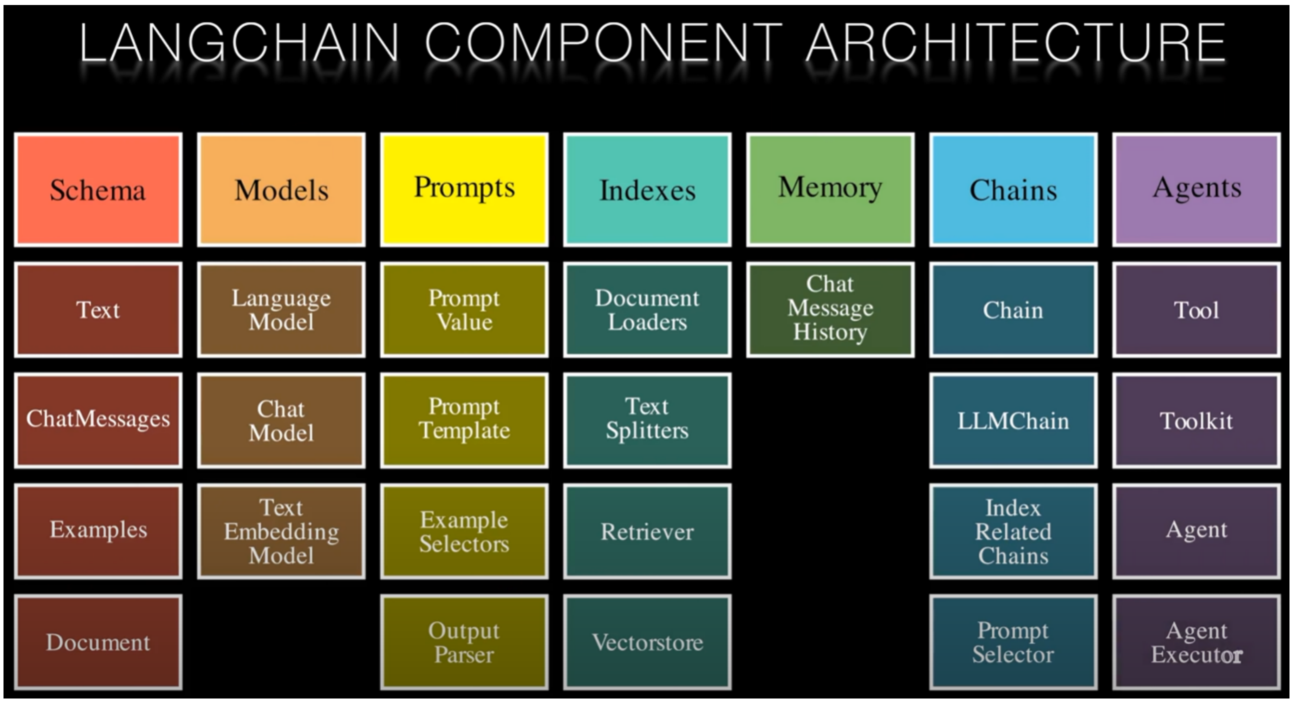
LangChain is a framework for developing applications powered by language models. The most powerful and differentiated applications will not only call out to a language model but will also be:

- *Data-aware*: connect a language model to other sources of data
- *Agentic*: allow a language model to interact with its environment

In this article, we might focus on the Agentic which are agents that are designed to be more long-running. You give them one or multiple long-term goals, and they independently execute those goals. The applications combine tool usage and long-term memory.

4.2.2 Langchain Concepts and Modules

These modules are the core abstractions that we view as the building blocks of any LLM-powered application.

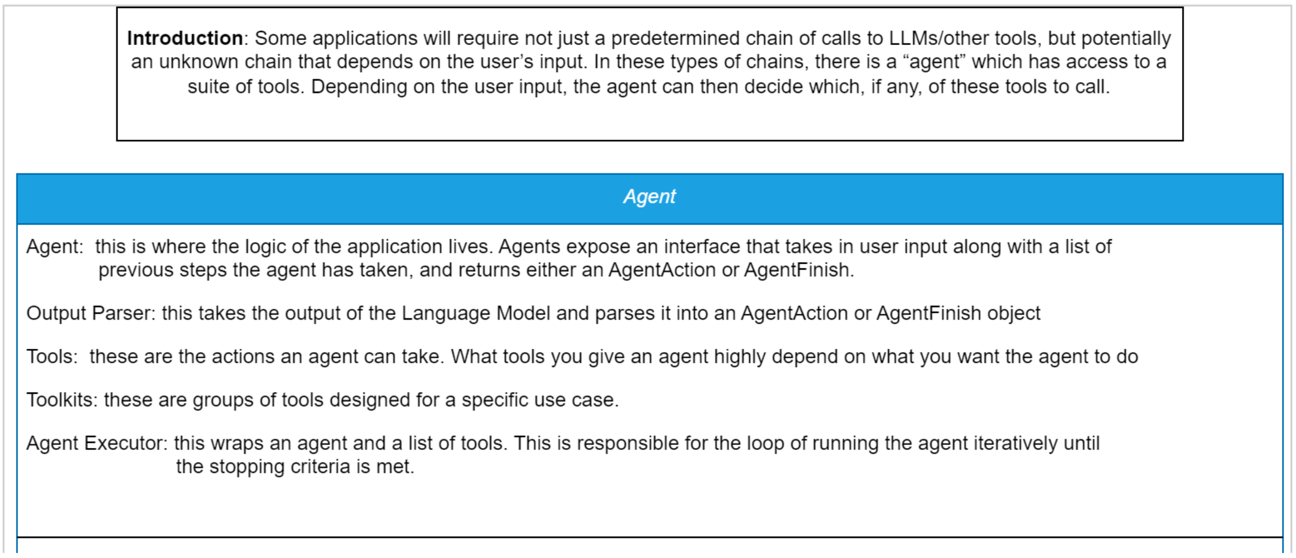


Component	Role	Description
Schema	Architect	* is a formal representation of the data types used in the system. * serves the Bedrock of Langchain providing structure and standardization and * enabling seamless interaction of different components.
Models	Engineer	* is the AI engine, responsible for text generation conversation and text embedding.
Prompts	Translator	* to understand inputs, generate meaningful outputs, and provide context for interactions. * are the interface that facilitates communication between the user and the models. * format user input and generate suitable prompts and parse the model output into a human-understandable format.

Indices	Librarian	* handles document management. * interact with Schema to manage categorize and store data. * integrate with models to retrieve and supply relevant information. * it ensures readily accessible.
Memory	Memory	* is primarily managed through the chat message history. * store pass interactions and can pass these to the models to provide context to the conversation.
Chains	Consultant/ Assistant	* end-to-end wrappers that combine various components to accomplish a common use case. * crucial for managing complex interactions operations and workflows. * orchestrating the interactions between the models, prompts, indices, and memory.
Agents	Manager	* are the action-oriented components of Langchain. * take user input, decide on an action use the appropriate tool, and generate a response. * agent works closely with other components to integrate them into a cohesive interactive and responsive AI system.

4.2.3 Flowchart Visualization

That are complex systems with large modules and concepts. I drew some flowcharts for visualization and easy understanding.



Introduction: Some applications will require not just a predetermined chain of calls to LLMs/other tools, but potentially an unknown chain that depends on the user's input. In these types of chains, there is a "agent" which has access to a suite of tools. Depending on the user input, the agent can then decide which, if any, of these tools to call.

Agent

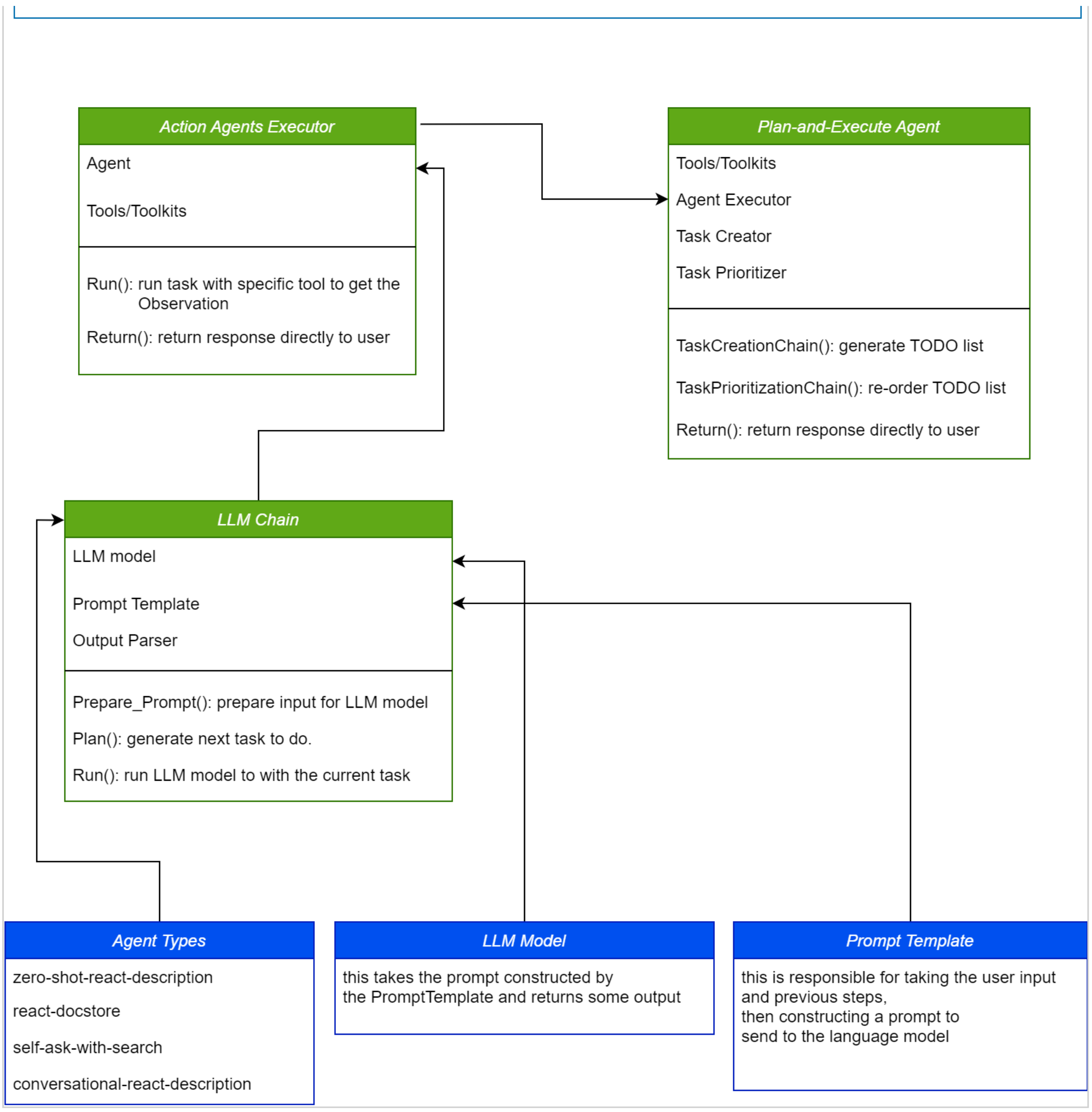
Agent: this is where the logic of the application lives. Agents expose an interface that takes in user input along with a list of previous steps the agent has taken, and returns either an AgentAction or AgentFinish.

Output Parser: this takes the output of the Language Model and parses it into an AgentAction or AgentFinish object

Tools: these are the actions an agent can take. What tools you give an agent highly depend on what you want the agent to do

Toolkits: these are groups of tools designed for a specific use case.

Agent Executor: this wraps an agent and a list of tools. This is responsible for the loop of running the agent iteratively until the stopping criteria is met.


Action Agents Executor
Agent
Tools/Toolkits
Run(): run task with specific tool to get the Observation
Return(): return response directly to user
Plan-and-Execute Agent
Tools/Toolkits
Agent Executor
Task Creator
Task Prioritizer
TaskCreationChain(): generate TODO list
TaskPrioritizationChain(): re-order TODO list
Return(): return response directly to user
LLM Chain
LLM model
Prompt Template
Output Parser
Prepare_Prompt(): prepare input for LLM model
Plan(): generate next task to do.
Run(): run LLM model to with the current task
Agent Types
zero-shot-react-description
react-docstore
self-ask-with-search
conversational-react-description
LLM Model
this takes the prompt constructed by the PromptTemplate and returns some output
Prompt Template
this is responsible for taking the user input and previous steps, then constructing a prompt to send to the language model

For detailed flowcharts, please refer to the below diagrams:

Action_Agent.drawio.pdf

Agent_Overview.drawio.pdf

Custom_Agent_Executor.drawio.pdf

Plan-and-Execute Agents.drawio.pdf

4.2.4 Conclusion

5 05_Data Preparation

5.1 [MCAL AI][Database guideline] Work Product Information (Wps.xlsx)

5.1.1 Introduction

The Wps.xlsx file is an essential document containing information about all the guidelines for RD - IT processes in the Autosar MCAL (Microcontroller Abstraction Layer) project. MCAL is a crucial component in embedded systems projects, and as its name suggests, it creates an abstraction layer between hardware and software, allowing software development independent of hardware.

File Content:

1. **Links to MCAL Work Products:** In the Wps.xlsx file, you will find a list of links to MCAL work products. These work products may include technical documents, source code, test suites, technical specifications, and any other materials relevant to MCAL development and deployment.
2. **Guideline Instruction Files:** Wps.xlsx contains a list of guideline instruction files, aiding in the implementation and development of MCAL according to Autosar standards. These files may encompass coding style guidelines, testing procedures, programming standards, and other design conventions.
3. **Introduction to MCAL Work products:** For each work product listed in the Wps.xlsx file, there is an accompanying introduction section, providing an overview of the functions, purposes, and usage of that specific work product. This ensures clear understanding and consensus when utilizing the MCAL work products.

Importance of Wps.xlsx:

1. **Centralized Information:** Wps.xlsx consolidates all the relevant information about MCAL guidelines, saving time and effort in searching for related information.
2. **Quality Assurance:** The guidelines in Wps.xlsx support the proper implementation of MCAL and adherence to Autosar standards, ensuring the quality and reliability of the final product.
3. **Convenience for Development Team:** All members of the development team can access and refer to Wps.xlsx, ensuring consistency in MCAL implementation.

Note: For accurate and comprehensive content from the Wps.xlsx file, it is advisable to directly review the document, as the description provided here is only a general introduction.

5.1.2 Scope

The scope of the Wps.xlsx file is to provide a comprehensive collection of guidelines and related work products for the RD - IT processes in the context of the Autosar MCAL (Microcontroller Abstraction Layer) project. The file aims to cover various aspects related to the development, implementation, and deployment of MCAL, ensuring adherence to the Autosar standards and best practices. The primary objectives and scope of the file are as follows:

1. **Guideline Coverage:** Wps.xlsx encompasses a wide range of guidelines that pertain to different stages and activities involved in the development of MCAL. These guidelines may include coding standards, testing procedures, architectural design principles, documentation conventions, and various other aspects essential for creating robust and reliable MCAL software.
2. **Work product Catalog:** The file serves as a catalog that consolidates links to all the work products relevant to the MCAL development process. These work products could be in the form of technical documents, source code, test suites, configuration files, and any other artifacts created during the development lifecycle.
3. **Consistency and Standardization:** By providing a set of guidelines, Wps.xlsx ensures consistency and standardization across the RD-IT processes within the MCAL project. It helps maintain a unified approach to software development, making it easier for developers to understand, collaborate, and deliver high-quality MCAL components.
4. **Quality Assurance:** The guidelines and instructions included in the file are designed to promote quality assurance practices. Following these guidelines helps prevent common coding errors, enhances software reliability, and reduces the likelihood of defects, thereby improving the overall quality of the MCAL implementation.
5. **Team Collaboration:** Wps.xlsx facilitates effective collaboration among the members of the RD - IT team and other stakeholders involved in the MCAL project. It provides a centralized resource where all team members can access and refer to essential guidelines and work products, fostering better communication and understanding.
6. **Project Governance:** The file plays a role in project governance by setting out the policies, procedures, and best practices that guide the MCAL development process. It helps project managers and team leads establish a well-structured development approach, monitor progress, and ensure compliance with Autosar standards.
7. **Documentation and Knowledge Sharing:** Wps.xlsx serves as a valuable documentation tool, capturing knowledge and lessons learned during the MCAL development. It facilitates knowledge sharing among team members and aids in onboarding new team members to the project efficiently.

Note: The specific scope and content of the Wps.xlsx file may vary depending on the project's requirements, development methodologies, and the organization's practices related to Autosar MCAL development.

5.1.3 Detail

There are 8 columns in Wps.xlsx:

PHA SE	Descripti on	Work Product	Overview Wps Description	File Name	Metho d	Link refer Guideline	Work product Location

PHASE: This column represents the phases in the development process, from Research and Development (RD) to Information Technology (IT). These phases reflect the necessary steps and activities to successfully develop MCAL software following the Autosar standard.

Description: This column provides a brief description of each phase. The description includes key contents, objectives, and main activities performed in that phase, helping readers understand the steps and roles of each phase in the MCAL development process.

Work Product: This column lists all the work products created during the development process. For each phase, it includes a list of documents, source code, test suites, and other relevant resources related to MCAL implementation.

File Name: This column contains the names of each work product listed in the "Work Product" column. The file names help identify and differentiate between different work products and enable team members to access resources easily.

Overview Wps Description: This column provides an overview description of the work products listed in the "Work Product" column. It explains the content and purpose of each work product, clarifies its role in the MCAL development process, and highlights important points to consider when using it.

Method: This column summarizes the methods and procedures used in each phase and related to specific work products. The method could be a guideline, a standard procedure, or practical instructions to execute activities accurately and efficiently.

Link refer Guideline: This column provides links to the guidelines related to each work product and method used in each phase. This helps team members access reference materials to work according to standards and ensure the quality of the product.

Work Product Location: This column provides information about the storage location of the work products during the MCAL project. This information helps pinpoint where the documents, source code, and related resources are stored, enabling the development team to easily find and access the necessary work products throughout the MCAL implementation. The details in this column may include folder paths, version control system details, online document system links, or other supplementary information depending on the project's storage structure and organization.

6 05_Knowledge_Transfer

6.1 General

This page serves as a centralized location to capture essential information and knowledge transfer details from team members. As part of our commitment to seamless transitions and effective knowledge sharing, this documentation will help ensure that valuable insights and expertise are transferred to the remaining team members.

6.2 Summarized knowledge

No	Assignee	Key Knowledge	Overview	Confluence links	Jira links	Source code	Output
1	Le Huu Minh Duc	Finetune model	• General knowledge • GG colab code • Instruction on how to create data • Instruction on how to finetune • Instruction on inference	[MCAL_AI] Finetune LLM Model[6]	ASDMCALCICD-86	Finetune[7]	
2	Le Huu Minh Duc	Create final data	• Explain the data file structure • Explain content structure • Code to create final data	[MCAL_AI] Create The Final Database[8]	ASDMCALCICD-93	Create_final_data[9]	final_d

6 https://jira.renesas.eu/confluence/display/ASDMCALCICD/%5BMCAL_AI%5D+Finetune+LLM+Model
7 https://renesasgroup.sharepoint.com/:f:/r/sites/team_MCAL_CICD_QI_AI/Shared%20Documents/Resources/03_AI/06_Model_Resources/Finetune?csf=1&web=1&e=2xLNKJ
8 https://jira.renesas.eu/confluence/display/ASDMCALCICD/%5BMCAL_AI%5D+Create+The+Final+Database
9 https://renesasgroup.sharepoint.com/:f:/r/sites/team_MCAL_CICD_QI_AI/Shared%20Documents/Resources/03_AI/06_Model_Resources/Create_final_data?csf=1&web=1&e=Tz9VVG

No	Assignee	Key Knowledge	Overview	Confluence links	Jira links	Source code	Output
							ata[10] model[11]
3	Le Huu Minh Duc	Question answer about a document	• Flow to retrieve data → Final prompt	[MCAL_AI] Question Answer about document[12]	ASDMCALCICD-100 ASDMCALCICD-91	MCAL_BOT[13]	final_data[14] model[15]
4	Ngo Quang Loc	Create middle data	• Explain data file structure • Explain content structure • Code to create final data	[MCAL_AI] Create The Middle Database[16]	ASDMCALCICD-92 ASDMCALCICD-90	ConvertScriptV4[17]	

10 https://renesasgroup.sharepoint.com/:f:/r/sites/team_MCAL_CICD_QI_AI/Shared%20Documents/Resources/03_AI/07_final_data?csf=1&web=1&e=U2lDu8
11 https://renesasgroup.sharepoint.com/:f:/r/sites/team_MCAL_CICD_QI_AI/Shared%20Documents/Resources/03_AI/06_Model_Resources/model?csf=1&web=1&e=7w3FYc
12 https://jira.renesas.eu/confluence/display/ASDMCALCICD/%5BMCAL_AI%5D+Question+Answer+about+document
13 https://renesasgroup.sharepoint.com/:f:/r/sites/team_MCAL_CICD_QI_AI/Shared%20Documents/Resources/03_AI/06_Model_Resources/MCAL_BOT?csf=1&web=1&e=RvLMCe
14 https://renesasgroup.sharepoint.com/:f:/r/sites/team_MCAL_CICD_QI_AI/Shared%20Documents/Resources/03_AI/07_final_data?csf=1&web=1&e=U2lDu8
15 https://renesasgroup.sharepoint.com/:f:/r/sites/team_MCAL_CICD_QI_AI/Shared%20Documents/Resources/03_AI/06_Model_Resources/model?csf=1&web=1&e=7w3FYc
16 https://jira.renesas.eu/confluence/display/ASDMCALCICD/%5BMCAL_AI%5D+Create+The+Middle+Database
17 https://renesasgroup.sharepoint.com/:f:/r/sites/team_MCAL_CICD_QI_AI/Shared%20Documents/Resources/03_AI/04_Convert_Script/ConvertScriptV4?csf=1&web=1&e=EHeloZ

No	Assignee	Key Knowledge	Overview	Confluence links	Jira links	Source code	Output
5	Ngo Quang Loc	Detect Figure	• Data collection • Label data • Training model	[AI - NLP] Detect Figure in Document (see page 37)		Detect Figure[18]	
6	Nguyen Duy Khanh	Run Web chat application	• How to get web chat resource • Resource structure • How to use	WebChat Starting Instructions (see page 83)		Web chat[19]	
7	La Nhat Hy	Baby-AGI Agent MetaPrompt Agent	• How to define tools for specific tasks. • How to use BabyAGI to do our task. • How to use MetaPrompt to improve the response.		ASDMCALCICD-68	[FINAL]_BabyAGI_MetaPrompt_Agents_on_multi_GPUs.ipynb[20]	

18 https://renesasgroup.sharepoint.com/:f:/r/sites/team_MCAL_CICD_QI_AI/Shared%20Documents/Resources/03_AI/04_Convert_Script/Detect%20Figure?csf=1&web=1&e=Q98j5D
19 https://renesasgroup.sharepoint.com/:f:/r/sites/team_MCAL_CICD_QI_AI/Shared%20Documents/Resources/03_AI/05_Web_Chat_Resources/Latest%20version?csf=1&web=1&e=Sm3HeX
20 https://renesasgroup.sharepoint.com/:u:/r/sites/team_MCAL_CICD_QI_AI/Shared%20Documents/Resources/03_AI/09_Autonomous_Agent/%5BFINAL%5D_BabyAGI_MetaPrompt_Agents_on_multi_GPUs.ipynb?csf=1&web=1&e=Plr86e

No	Assignee	Key Knowledge	Overview	Confluence links	Jira links	Source code	Output
8	La Nhat Hy	Customized Agent	• Instructor Agent for detailed instructed tasks. • Task Spec Agent for pre-defined tasks.		Instructor Agent: ASDMCALCICD-101 Task Spec Agent: ASDMCALCICD-102	[FINAL]_Task_Spec_Agent_Jira.ipynb[21]	

6.3 Q&A

To clarify the knowledge, please support resolving the concern items in the below table

No	Assignee	Specification	Question	Answer

1	@Duc Le-minh [RVC_BV]	MCAL_BOT/_saver.py
		`def save_TFIDF_vector(variable, path):`
		`    vectorizer = TfidfVectorizer()`
		`    # Save the vectorizer`
		`    vectorizer_path = path + '/vectorizer_TFIDF.pkl'`
		`    with open(vectorizer_path, 'wb') as file:`
		`        pickle.dump(vectorizer, file)`
			Why do we need to save Tokenizer as a pickle file?	We need to save Tokenizer as pickle file because it takes to much time to load documents to tokenizer. So that if we save the tokenizer, we can load it very fast.
				Why .pkl file? Because .pkl file support saving Python variable

21 https://renesasgroup.sharepoint.com/:u:/r/sites/team_MCAL_CICD_QI_AI/Shared%20Documents/Resources/03_AI/09_Autonomous_Agent/%5BFINAL%5D_Task_Spec_Agent_Jira.ipynb?csf=1&web=1&e=qpvwhU

No	Assignee	Specification	Question	Answer

2	@Duc Le-minh [RVC_BV]	MCAL_BOT/_retrieve.py
		`## Thresholding the scores`
		`filtered_similarity_scores = [score for score in similarity_scores if score > self.threshold]`
		`## Thresholding the indices`
		`filtered_match_indices = [index for index, score in zip(match_indices, similarity_scores) if score > self.threshold]`
			For retrieving information, why the threshold for cutting down indices is 0.0 value?
			Should it be a value greater than zero?
				For some questions like: Hello, How are you, ... the score is equal to zero. I just want to remove the data retrieve for these questions.
				You can try to increase the threshold value to evaluate the result. I have not tried it yet.

3	@Duc Le-minh [RVC_BV]	MCAL_BOT/_retrieve.py
		`def get_index_data(self, document, number_of_query):`
		`    top_match_indices = sorted(range(len(similarity_scores)), key=lambda i: similarity_scores[i], reverse=True)`
			Extracted from the sorted document list, so the similarity_scores list is already sorted too, Why do we need to sort again to get the top indices list?	The first sort is to find the best match of data in each file. For each file I have a top list. When we retrieve through many files, like in a folder, I will combine them to get the total top list. This top list is quite big so I have to sort the total top list (second sort) and get the final top list (top indices list)

6.4 Conclusion

By completing this documentation, outgoing team members play a vital role in ensuring a successful transition. The information they provide will be invaluable to the remaining team members, enabling them to continue projects smoothly and effectively. Thank you for your dedication to knowledge sharing and your contribution to the transition process.