Human Evaluation on DALLE

This form is intended to evaluate the quality of the results produced by DALLE, an LLM-based assistant designed to extract microservice-based architecture designs from textual input.

Overview of the Evaluation

You will be asked to evaluate 3 *different projects*. Each project includes:

- A *brief description* of the system to be generated
- A set of *user stories* related to the system

These inputs are provided to the DALLE system, which then produces an architectural design. Among others outputs, DALLE identifies:

- The **list of microservices**
- The **microservice and communication patterns** (based on the reference book)

Evaluation Process

For each project, you will be presented with:

- The original *system description* and *user stories*;
- A visual representation of the architecture generated by DALLE, including microservice patterns.

You are asked to evaluate whether the identified **data and communication style patterns** are appropriate and consistent with the project requirements.

Each evaluation question uses a **Likert scale from 1 to 7**, where:

- 1 = Strongly Disagree
- 7 = Strongly Agree

You may also provide **additional comments or observations** for each project in the dedicated Observation section.

Reference Patterns

The evaluation is based on the following architecture design patterns (as defined in the reference book):

Communication Style Patterns:

- Shared Database
- Database per Service

Data Style Patterns:

- API Composition

- CQRS
- Saga
- Aggregate
- Event Sourcing
- Domain Event

Final Questions
At the end of the form, you will find a few general questions about the perceived utility and effectiveness of DALLE as a tool for architectural design support.

* Indicates required question

1. Rate your confidence in designing systems based on microservices architectures: *

1	2	3	4	5
☆	☆	☆	☆	☆

2. How many years of experience do you have? *

3. What is your age? *

4. In what gender do you identify yourself? *

Mark only one oval.

- ◯ Male
- ◯ Female
- ◯ Prefer not to say

5. Have you ever evaluated systems' architectures? *

Mark only one oval.

- Yes
- No

6. Describe your role in the SE field *

Mark only one oval.

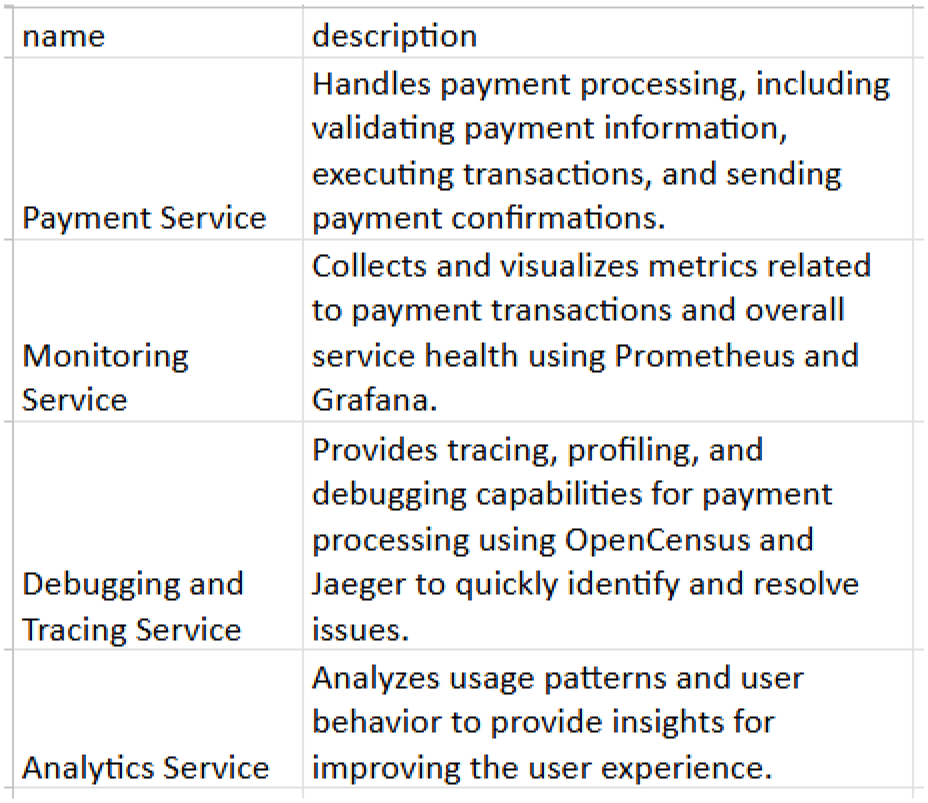
- Practitioner
- Academic

Hipstershop

SYSTEM DESCRIPTION
This microservice application is a demo designed to showcase a payment service within a larger ecosystem of microservices. It utilizes key technologies such as Kubernetes for orchestration, gRPC for inter-service communication, Istio for service mesh capabilities, and OpenCensus for tracing and monitoring. The application is instrumented for performance profiling, tracing, and debugging, and it integrates with tools like Prometheus and Grafana for metrics visualization, as well as Jaeger for distributed tracing.

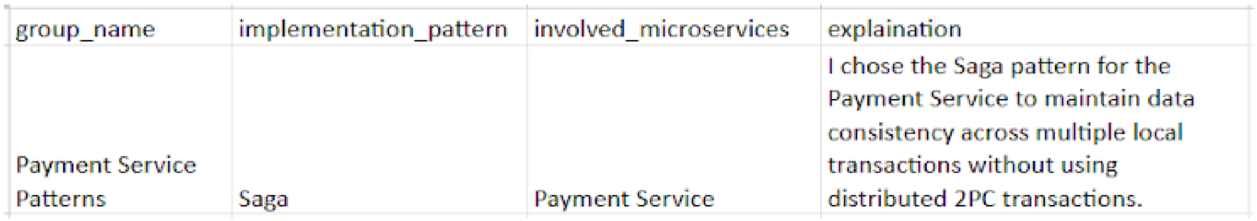
USER STORIES
- 1. As a customer, I want to make a payment so that I can complete my purchase.
- 2. As a developer, I want to monitor payment transactions so that I can ensure the service is functioning correctly.
- 3. As a system administrator, I want to view performance metrics so that I can optimize the application’s performance.
- 4. As a user, I want to receive confirmation of my payment so that I know my transaction was successful.
- 5. As a developer, I want to debug payment processing issues so that I can resolve them quickly.
- 6. As a customer, I want to validate my payment information so that I can avoid errors during the transaction.
- 7. As a product manager, I want to analyze usage patterns so that I can improve the user experience.

Microservices



name	description
Payment Service	Handles payment processing, including validating payment information, executing transactions, and sending payment confirmations.
Monitoring Service	Collects and visualizes metrics related to payment transactions and overall service health using Prometheus and Grafana.
Debugging and Tracing Service	Provides tracing, profiling, and debugging capabilities for payment processing using OpenCensus and Jaeger to quickly identify and resolve issues.
Analytics Service	Analyzes usage patterns and user behavior to provide insights for improving the user experience.

Patterns



group_name	implementation_pattern	involved_microservices	explaination
Payment Service Patterns	Saga	Payment Service	I chose the Saga pattern for the Payment Service to maintain data consistency across multiple local transactions without using distributed 2PC transactions.

7-point likert scale

1. Strongly disagree
2. Disagree
3. Somehow disagree
4. Neither agree nor disagree
5. Somehow agree
6. Agree
7. Strongly agree

7. Is the division in microservice correct with respect to the context and the user stories? *

1	2	3	4	5	6	7
☆	☆	☆	☆	☆	☆	☆

8. Is **Saga Pattern** correctly applied? *

1	2	3	4	5	6	7
☆	☆	☆	☆	☆	☆	☆

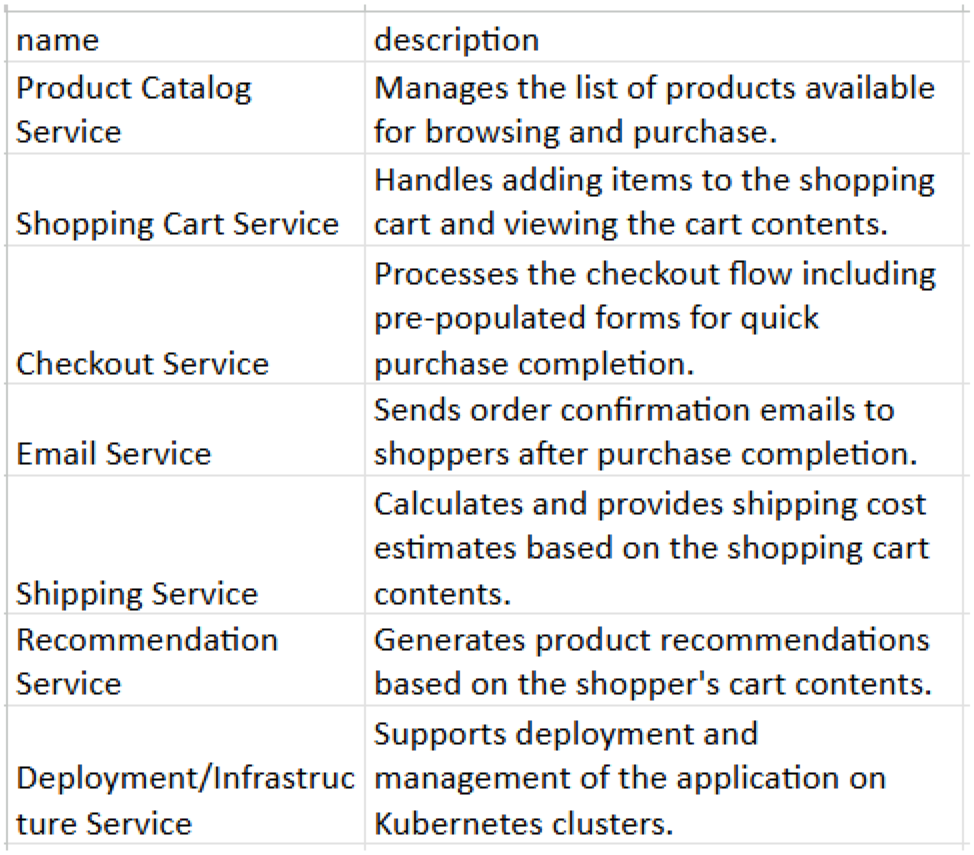
9. Observations

ServiceBoutique

SYSTEM DESCRIPTION
This microservice application is an e-commerce demo called Online Boutique, designed to showcase how developers can modernize enterprise applications using Google Cloud products. It allows users to browse items, add them to a shopping cart, and complete purchases. The application is built using 11 microservices, each developed in different programming languages, and communicates over gRPC. Key technologies include Google Kubernetes Engine (GKE), Cloud Service Mesh (CSM), gRPC, and various Google Cloud services like Spanner and Memorystore.

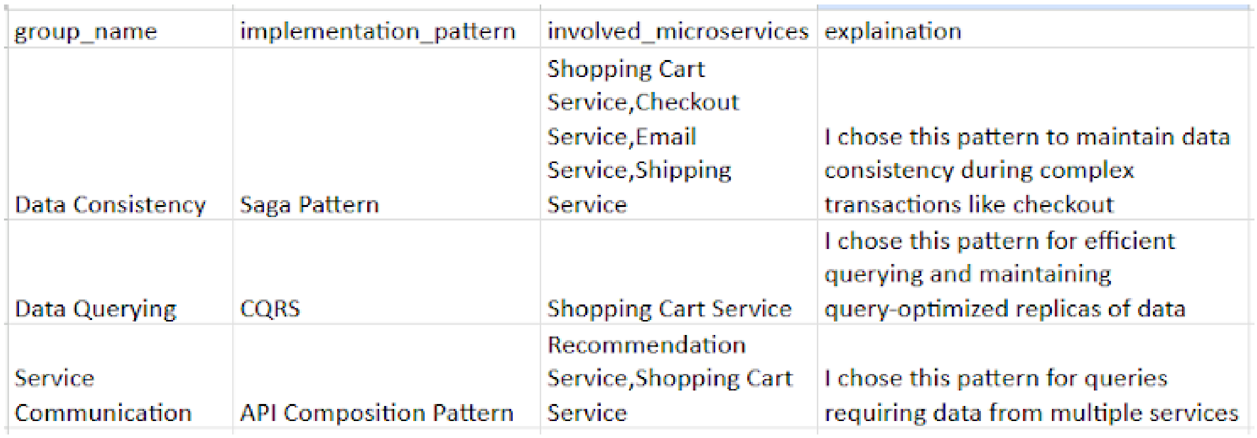
USER STORIES
- 1. As a shopper, I want to browse a list of products so that I can find items I want to purchase.
- 2. As a shopper, I want to add items to my shopping cart so that I can review my selections before buying.
- 3. As a shopper, I want to view my shopping cart so that I can see the items I have selected.
- 4. As a shopper, I want to check out using a pre-populated form so that I can complete my purchase quickly.
- 5. As a shopper, I want to receive an order confirmation email so that I have a record of my purchase.
- 6. As a shopper, I want to see shipping cost estimates based on my cart so that I can understand the total cost before checking out.
- 7. As a shopper, I want to receive product recommendations based on my cart so that I can discover related items I might like.
- 8. As a developer, I want to deploy the application on a Kubernetes cluster so that I can test and demonstrate its functionality.

Microservices



name	description
Product Catalog Service	Manages the list of products available for browsing and purchase.
Shopping Cart Service	Handles adding items to the shopping cart and viewing the cart contents.
Checkout Service	Processes the checkout flow including pre-populated forms for quick purchase completion.
Email Service	Sends order confirmation emails to shoppers after purchase completion.
Shipping Service	Calculates and provides shipping cost estimates based on the shopping cart contents.
Recommendation Service	Generates product recommendations based on the shopper's cart contents.
Deployment/Infrastructure Service	Supports deployment and management of the application on Kubernetes clusters.

Patterns



group_name	implementation_pattern	involved_microservices	explaination
Data Consistency	Saga Pattern	Shopping Cart Service,Checkout Service,Email Service,Shipping Service	I chose this pattern to maintain data consistency during complex transactions like checkout
Data Querying	CQRS	Shopping Cart Service	I chose this pattern for efficient querying and maintaining query-optimized replicas of data
Service Communication	API Composition Pattern	Recommendation Service,Shopping Cart Service	I chose this pattern for queries requiring data from multiple services

7-point likert scale

1. Strongly disagree
2. Disagree
3. Somehow disagree
4. Neither agree nor disagree
5. Somehow agree
6. Agree
7. Strongly agree

10. Is the division in microservice correct with respect to the context and the user stories? *

1	2	3	4	5	6	7
☆	☆	☆	☆	☆	☆	☆

11. Is **SAGA Pattern** correctly applied? *

1	2	3	4	5	6	7
☆	☆	☆	☆	☆	☆	☆

12. Is **CQRS Pattern** correctly applied? *

1	2	3	4	5	6	7
☆	☆	☆	☆	☆	☆	☆

13. Is **API Composition Pattern** correctly applied? *

1	2	3	4	5	6	7
☆	☆	☆	☆	☆	☆	☆

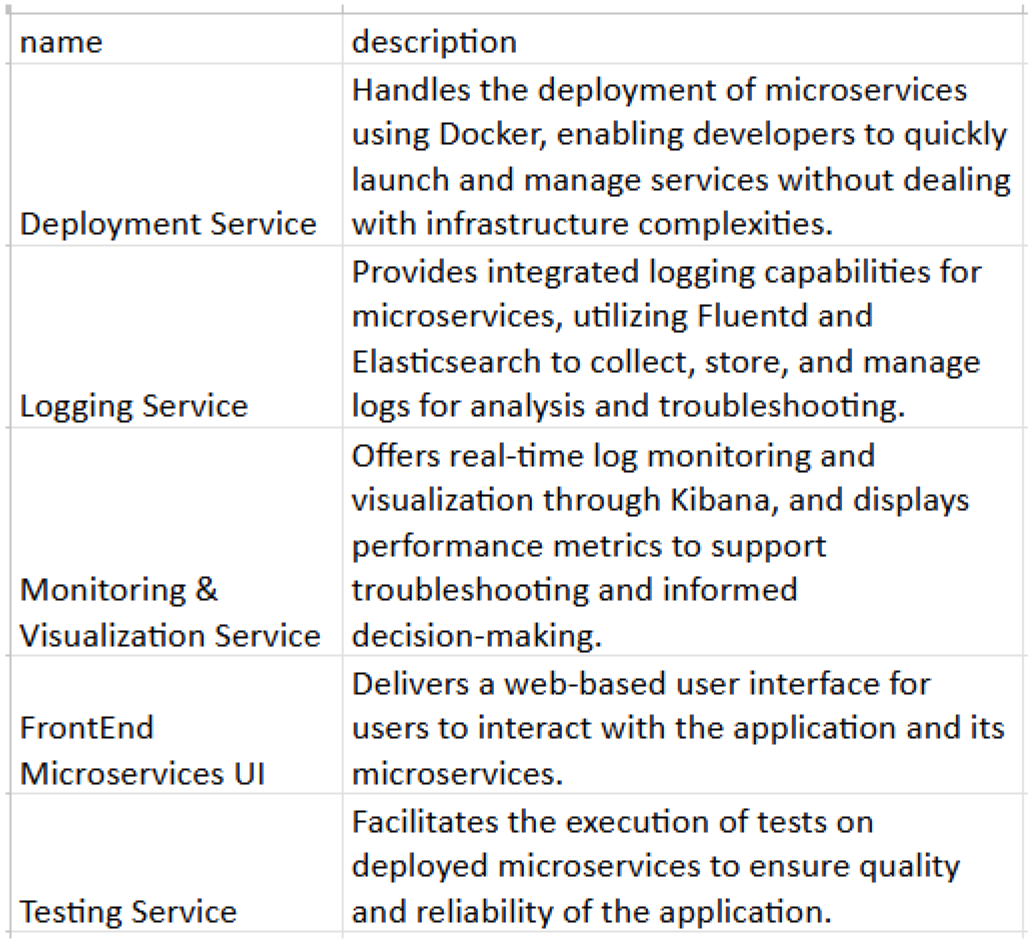
14. Observations

Rideshare

SYSTEM DESCRIPTION
This microservice application repository is designed to deploy the AIST Microservices Sandbox, which facilitates the development and testing of microservices. The main purpose is to provide a configurable environment for running microservices with integrated logging capabilities using Elasticsearch and Fluentd. Key technologies used include Docker for containerization, Kubernetes for orchestration, and EFK (Elasticsearch, Fluentd, Kibana) for logging and monitoring.

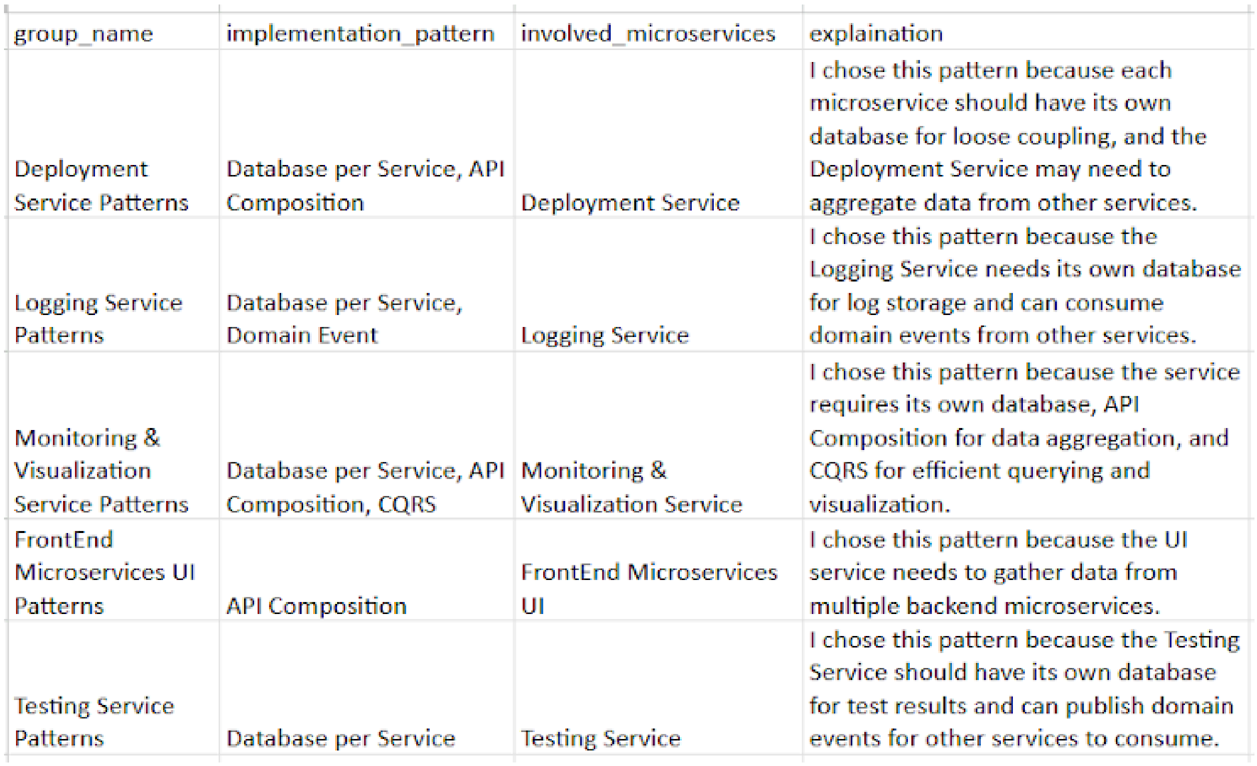
USER STORIES
- 1. As a developer, I want to deploy microservices quickly using Docker so that I can focus on building features without worrying about the underlying infrastructure.
- 2. As a system administrator, I want to monitor logs in real-time using Kibana so that I can troubleshoot issues effectively.
- 3. As a user, I want to access the FrontEnd Microservices web UI so that I can interact with the application easily.
- 4. As a developer, I want to configure logging for my microservices so that I can track and analyze application behavior.
- 5. As a QA engineer, I want to run tests on the deployed microservices so that I can ensure the application meets quality standards.
- 6. As a product owner, I want to visualize the performance metrics of the microservices so that I can make informed decisions about scaling and improvements.

Microservices



name	description
Deployment Service	Handles the deployment of microservices using Docker, enabling developers to quickly launch and manage services without dealing with infrastructure complexities.
Logging Service	Provides integrated logging capabilities for microservices, utilizing Fluentd and Elasticsearch to collect, store, and manage logs for analysis and troubleshooting.
Monitoring & Visualization Service	Offers real-time log monitoring and visualization through Kibana, and displays performance metrics to support troubleshooting and informed decision-making.
FrontEnd Microservices UI	Delivers a web-based user interface for users to interact with the application and its microservices.
Testing Service	Facilitates the execution of tests on deployed microservices to ensure quality and reliability of the application.

Patterns



group_name	implementation_pattern	involved_microservices	explaination
Deployment Service Patterns	Database per Service, API Composition	Deployment Service	I chose this pattern because each microservice should have its own database for loose coupling, and the Deployment Service may need to aggregate data from other services.
Logging Service Patterns	Database per Service, Domain Event	Logging Service	I chose this pattern because the Logging Service needs its own database for log storage and can consume domain events from other services.
Monitoring & Visualization Service Patterns	Database per Service, API Composition, CQRS	Monitoring & Visualization Service	I chose this pattern because the service requires its own database, API Composition for data aggregation, and CQRS for efficient querying and visualization.
FrontEnd Microservices UI Patterns	API Composition	FrontEnd Microservices UI	I chose this pattern because the UI service needs to gather data from multiple backend microservices.
Testing Service Patterns	Database per Service	Testing Service	I chose this pattern because the Testing Service should have its own database for test results and can publish domain events for other services to consume.

7-point likert scale

1. Strongly disagree
2. Disagree
3. Somehow disagree
4. Neither agree nor disagree
5. Somehow agree
6. Agree
7. Strongly agree

15. Is the division in microservice correct with respect to the context and the user stories? *

1	2	3	4	5	6	7
☆	☆	☆	☆	☆	☆	☆

16. Is **Saga Pattern** correctly applied? *

1	2	3	4	5	6	7
☆	☆	☆	☆	☆	☆	☆

17. Is **Database per Service Pattern** correctly applied? *

1	2	3	4	5	6	7
☆	☆	☆	☆	☆	☆	☆

18. Is **API Composition Pattern** correctly applied? *

1	2	3	4	5	6	7
☆	☆	☆	☆	☆	☆	☆

19. Is **CQRS Pattern** correctly applied? *

1	2	3	4	5	6	7
☆	☆	☆	☆	☆	☆	☆

20. Observations

____

____

____

____

____

Utility of DALLE

Please answer the following questions related to the utility of the proposed approach. Look at the demo of the tool to understand how it works.

21. Would you consider using DALLE to help design system architectures? *

Mark only one oval.

- Yes
- No
- Other: ____________________

22. Do you think DALLE could make your work easier or more efficient? *

Mark only one oval.

- Yes
- No
- Other: ____________________

23. Would you use DALLE to check or validate your design ideas? *

Mark only one oval.

- Yes
- No
- Other: ____________________

24. Do you find the suggestions provided by DALLE useful? *

Mark only one oval.

- Yes
- No
- Other: ____________________

This content is neither created nor endorsed by Google.

Google Forms