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

Train Ticket

SYSTEM DESCRIPTION
This microservice application is a train ticket booking system designed to facilitate the purchase and management of train tickets. It is built using a microservice architecture that comprises 41 microservices, leveraging various programming languages and frameworks, including Java with Spring Boot and Spring Cloud, Node.js with Express, Python with Django, and Go with Webgo. The system utilizes MongoDB and MySQL for database management, ensuring robust data handling and storage.

USER STORIES
- 1. As a traveler, I want to search for available train tickets so that I can find suitable travel options.
- 2. As a user, I want to book a train ticket so that I can secure my travel plans.
- 3. As a customer, I want to view my booking history so that I can keep track of my past travels.
- 4. As a user, I want to cancel my train ticket so that I can manage my travel plans effectively.
- 5. As a traveler, I want to receive notifications about my train schedule so that I am informed of any changes.
- 6. As a user, I want to pay for my ticket online so that I can complete my booking conveniently.
- 7. As an administrator, I want to manage train schedules so that I can ensure accurate information is available to users.
- 8. As a customer support agent, I want to access user inquiries so that I can assist customers with their issues.
- 9. As a user, I want to filter search results by price and time so that I can find the best options for my needs.
- 10. As a traveler, I want to provide feedback on my travel experience so that I can help improve the service.

Microservices

name	description
Ticket Search Service	Handles searching for available train tickets and filtering search results by various criteria such as price and time.
Booking Service	Manages the booking of train tickets, including creating new bookings and viewing booking history.
Cancellation Service	Allows users to cancel their train tickets and manage their travel plans effectively.
Notification Service	Sends notifications to travelers about their train schedules and any changes.
Payment Service	Handles online payment processing for ticket bookings.
Schedule Management Service	Enables administrators to manage train schedules to ensure accurate and up-to-date information.
Customer Support Service	Provides access to user inquiries for customer support agents to assist customers.
Feedback Service	Allows travelers to provide feedback on their travel experience to help improve the service.

Patterns

group_name	implementation_pattern	involved_microservices	explaination
Distributed Transactions	Saga	Booking Service,Payment Service,Cancellation Service	I chose this pattern to manage distributed transactions across services like Booking, Payment, and Cancellation, ensuring eventual consistency and avoiding two-phase commits.
Data Querying	API Composition	Ticket Search Service,Booking Service,Schedule Management Service	I chose this pattern to efficiently query data from multiple services and aggregate results for user queries.

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 **API Composition Pattern** correctly applied? *

1 2 3 4 5 6 7

☆ ☆ ☆ ☆ ☆ ☆ ☆

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

1 2 3 4 5 6 7

☆ ☆ ☆ ☆ ☆ ☆ ☆

10. Observations

Sock Shop

SYSTEM DESCRIPTION
This microservice application serves as a reference demo for microservices, showcasing best practices and common pitfalls. Its main purpose is to provide a testable application that demonstrates the integration of microservices within an ECommerce context, emphasizing continuous integration and deployment. The architecture is designed to be polyglot, utilizing various technologies to illustrate different aspects of microservices. Communication between services is facilitated through REST over HTTP.

USER STORIES
- 1. As a customer, I want to browse products so that I can find items to purchase.
- 2. As a customer, I want to add items to my shopping cart so that I can review my selections before checkout.
- 3. As a customer, I want to complete my purchase so that I can acquire the products I want.
- 4. As a system administrator, I want to monitor service performance so that I can ensure the application runs smoothly.
- 5. As a developer, I want to deploy new features quickly so that I can enhance the application without downtime.
- 6. As a tester, I want to validate the application functionality so that I can ensure a seamless user experience.
- 7. As a user, I want to receive notifications about my order status so that I am informed about my purchases.

Microservices

name	description
Product Catalog Service	Handles browsing and retrieval of product information for customers.
Shopping Cart Service	Manages the addition and review of items in a customer's shopping cart.
Order Management Service	Processes order completion and manages purchase transactions.
Notification Service	Sends notifications to users about their order status and updates.
Monitoring Service	Monitors the performance and health of all microservices to ensure smooth operation.
Deployment Service	Facilitates continuous integration and deployment of new features with minimal downtime.
Testing Service	Validates application functionality to ensure a seamless user experience.

Patterns

group_name	implementation_pattern	involved_microservices	explaination
Product Catalog Service Pattern	Database per Service	Product Catalog Service	I chose this pattern to maintain the product data store independently for loose coupling.
Shopping Cart Service Pattern	Saga	Shopping Cart Service,Order Management Service	I chose this pattern for managing distributed transactions between cart and order processing.
Order Management Service Pattern	Saga	Order Management Service	I chose this pattern to coordinate order completion steps across 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

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

1 2 3 4 5 6 7

☆ ☆ ☆ ☆ ☆ ☆ ☆

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

1 2 3 4 5 6 7

☆ ☆ ☆ ☆ ☆ ☆ ☆

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

1 2 3 4 5 6 7

☆ ☆ ☆ ☆ ☆ ☆ ☆

14. Observations

TeaStore

SYSTEM DESCRIPTION
The application is a micro-service reference and test platform designed for benchmarking and testing purposes. It simulates a web store that sells tea and tea supplies, featuring user interface elements for database generation and service resetting. The architecture consists of five distinct services plus a registry, allowing for unlimited replication and deployment across various devices. Communication between services is facilitated through REST and the Netflix Ribbon client-side load balancer. Additionally, pre-instrumented variants of the services utilize Kieker for detailed monitoring of the application's actions and behavior.

USER STORIES
- 1. As a customer, I want to browse available tea products so that I can find items to purchase.
- 2. As a customer, I want to add items to my shopping cart so that I can review my selections before checkout.
- 3. As a customer, I want to complete my purchase securely so that I can receive my tea supplies.
- 4. As an administrator, I want to reset the service so that I can clear any test data and start fresh.
- 5. As a developer, I want to generate load on the application so that I can test its performance under stress.
- 6. As a tester, I want to instrument the application with monitoring tools so that I can analyze its behavior and performance metrics.
- 7. As a user, I want to deploy the application using Docker Compose so that I can easily manage its services.
- 8. As a user, I want to run the application on a Kubernetes cluster so that I can leverage orchestration features for scalability.

Microservices

name	description
Product Catalog Service	Manages and provides information about available tea products for browsing.
Shopping Cart Service	Handles adding items to the shopping cart and managing user selections before checkout.
Order Processing Service	Processes and completes customer purchases securely, managing transactions and order fulfillment.
Service Reset Service	Allows administrators to reset services, clearing test data and restoring initial state.
Load Generation Service	Generates load on the application to test performance under stress conditions.
Monitoring and Instrumentation Service	Provides instrumentation using Kieker for detailed monitoring and analysis of application behavior and performance.
Deployment Orchestration Service	Supports deployment and management of the application using Docker Compose and Kubernetes for scalability and orchestration.

Patterns

group_name	implementation_pattern	involved_microservices	explaination
Data Consistency	Saga	Shopping Cart Service,Order Processing Service	I chose this pattern to manage distributed transactions that span adding items to the cart and order processing, ensuring data consistency without distributed transactions.
Data Isolation	Database per Service	Product Catalog Service,Shopping Cart Service	I chose this pattern to ensure loose coupling by having separate databases for each service.

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. 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.

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

Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Other: _____

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

Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Other: _____

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

Mark only one oval.

- Yes
- No
- Other: _____

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

Mark only one oval.

- Yes
- No
- Other: _____