Assignment 3: Strangler fig pattern for an e-commerce website

Assignment for an architect to implement the strangler fig pattern by adding an anti-corruption layer to a monolithic application and transitioning to a microservices architecture for an e-commerce website:

Scenario:

You work for an e-commerce company that has been using a monolithic application for a long time. However, due to scalability and maintenance issues, the company has decided to transition to a microservices architecture. The current application has various modules for order management, product catalog, customer management, payment, and shipping. You are responsible for implementing the strangler fig pattern to gradually transition to a microservices architecture.

Assignment:

1. Perform a thorough analysis of the existing monolithic application to identify the key modules that can be broken down into individual microservices.
2. Create a detailed plan to create an anti-corruption layer for the existing application that can interact with the new microservices. This layer should be created using a separate API gateway that can handle all incoming requests and delegate them to the appropriate microservices.
3. Develop and deploy microservices for each identified module using the appropriate technology stack and design patterns.
4. Gradually migrate the existing codebase to use the anti-corruption layer and the new microservices. This should be done by gradually removing dependencies on the monolithic application and replacing them with calls to the API gateway and the corresponding microservices.
5. Implement automated testing for the new microservices and the anti-corruption layer to ensure that they are functioning correctly and providing the expected results.
6. Finally, monitor the new microservices and the API gateway to ensure that they are working efficiently and providing optimal performance.

# Example:

Assuming that the existing monolithic application has a module for product catalog management that can be broken down into individual microservices, the following steps can be taken to implement the strangler fig pattern:

1. Create a product catalog microservice that can handle all requests related to product management, such as adding new products, updating prices, and managing inventory.
2. Create an anti-corruption layer in the form of an API gateway that can handle all incoming requests related to product catalog management and delegate them to the new product catalog microservice.
3. Gradually remove all references to the monolithic product catalog module in the existing codebase and replace them with calls to the new API gateway and microservice.
4. Test the new microservice and the anti-corruption layer using automated tests to ensure that they are functioning correctly and providing the expected results.
5. Monitor the new microservice and the API gateway to ensure that they are working efficiently and providing optimal performance.

Solution:

1. Catalog Service API - This API would be responsible for managing the product catalog for the ecommerce website. It would allow users to view, search, and filter products, as well as add them to their cart.
2. Order Service API - This API would be responsible for managing the ordering process for the ecommerce website. It would allow users to place orders, view their order history, and track the status of their orders.
3. Payment Service API - This API would be responsible for handling payments for the ecommerce website. It would allow users to add and manage their payment methods, as well as process payments for their orders.
4. User Service API - This API would be responsible for managing user accounts for the ecommerce website. It would allow users to create and manage their accounts, update their profile information, and view their order history.
5. Cart Service API - This API would be responsible for managing the user's cart for the ecommerce website. It would allow users to add and remove items from their cart, view the contents of their cart, and initiate the checkout process.
6. Shipping Service API - This API would be responsible for managing the shipping process for the ecommerce website. It would allow users to select their shipping preferences, view the status of their shipments, and track their packages.

These APIs could be implemented using a microservices architecture, with each service responsible for a specific set of functions related to the ecommerce website. The anti-corruption layer would be used to translate requests from the monolith application into requests that can be processed by the microservices. Over time, as more and more functionality is migrated to the microservices, the monolith can be gradually decommissioned.