**Converting Monoliths to Microservices**

A monolith application is an the one in which functions are tightly coupled with the data access codes and user interface integrated to form a single program (Mishra et al. 2018). The application consists of a database, client-side user interface, and server-side application combined into a single executable program. The primary challenges of monolith applications include limited reuse and difficulties in achieving operational agility. Additionally, scaling monolith applications can be challenging because developers must build updated versions to make changes to the system. Contrarily, microservice architecture delivers scalability and flexibility in the development of service-based applications since the architecture enables each service to be responsible for a specific part of the functionality. However, enabling monolith applications to operate as microservice can be challenging since microservice architecture has more granularity. For instance, for a monolith application to be enabled to function as a microservice, one must identify the domains and subdomains of the application and understand the dependencies within the codebase (Faustino et al. 2022). Another challenge in enabling monolith applications to function as microservice is finding shared mutable data. This is particularly important since shared mutable data must be modelled as a microservice in the monolith-microservice architecture migration (Kazanavicius et al. 2022). Additionally, static data like supported currencies and country codes in a monolith e-commerce application can be slow to manage or change thus creating further challenges in enabling monolith application to function as a microservice.

A monolith application like an e-commerce application can be enabled to function as a microservice by duplicating static data for different modules or services to their databases to be modelled as separate microservices. In addition, monolith applications can be enabled to function as a microservice through code analysis to discover monolithic APIs and group the API endpoints according to their resources (Janes and Russo 2019). In this approach, the group endpoints can be designed as a microservice. With these developments, the e-commerce monolith application example would function as a microservice by allowing the Order, Shipping and Payment microservices to utilize the APIs of the customer Shopping Status microservice, since services exist independent of others in microservices architecture.

**Reference List**

Faustino, D., Gonçalves, N., Portela, M. and Silva, A.R., 2022. Stepwise Migration of a Monolith to a Microservices Architecture: Performance and Migration Effort Evaluation. *arXiv preprint arXiv:2201.07226*.

Kazanavičius, J., Mažeika, D. and Kalibatienė, D., 2022. An Approach to Migrate a Monolith Database into Multi-Model Polyglot Persistence Based on Microservice Architecture: A Case Study for Mainframe Database. *Applied Sciences*, *12*(12), p.6189.

Mishra, M., Kunde, S. and Nambiar, M., 2018, September. Cracking the monolith: Challenges in data transitioning to cloud native architectures. In *Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings* (pp. 1-4).

Janes, A. and Russo, B., 2019, October. Automatic performance monitoring and regression testing during the transition from monolith to microservices. In *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)* (pp. 163-168). IEEE.