Statement of Design Selection

FDM06

Contents

1. Introduction

In FDM05, two technical designs have been proposed (Design A and Design B) based on specific design strategies (Strategy A and Strategy B, respectively). Each strategy uses different metrics and ideas to build a technical design proposal, to fulfil technical requirements while abiding by chosen design goals. This statement will compare the resulting design proposals and select the most appropriate, based on their ability to meet the design goals.

2. Reliability

Design A makes use of mature technology in relational databases, which brings reliability through continued support and maintenance of the tooling. However, the design attempts to implement a custom solution for encryption, where the field of cryptography is huge, and it would be difficult to ensure the implementation is secure enough for sensitive data. Finally, much of the code is designed with testability in mind, as many of the classes within the codebase are highly cohesive to improve the validity and implementation of unit tests. Many of the classes in the codebase are highly cohesive, to improve focus of unit testing and much of the code is built with testability in mind.

Design B uses separate databases and makes use of microservices, both of which can be containerized and thus have potential for distribution across many physical or virtual servers. Containerization would allow for many instances of a service to run, supporting load balancing and service failure. Therefore, this design contains more supporting architecture to implement many quality-of-service features in future.

3. Reusability

Design B and its microservice architecture is built to simulate object-oriented architecture, enabling the use of utility code throughout services to create reusable configuration throughout many services.

However, design A makes use of OOP principles and patterns such as interfaces and the repository pattern, which ensure maximum reusability throughout the monolithic codebase. Therefore, this design has much more supporting architecture than design B benefiting reusability in potential future development.

4. Maintainability

Design A makes use of a singular relational database with ACID guarantees, which ensures all data is always synchronized. Design B falls short in this regard, as NoSQL does not support this feature, creating extra steps when making changes to data. The codebase is well segregated into multiple cohesive components which reduce many difficulties when maintaining a monolithic codebase, which usually runs the risk of being tightly coupled. A REST API layer exists to separate business logic from controllers and routing, which reduces coupling in the code. This is similar to the API gateway in design B but provides much less of the benefits. The gateway creates a layer of obscurity to the back-end system, for an easier interface with the client codebase.

Design B's data model contains a non-rigid schema which poses negatives and positives for maintainable code. On the other hand, it is simple to scale and expand by changing the schema or introducing new databases. The API gateway makes use of the service registry pattern, making the network architecture more maintainable as making changes are simple and manageable.

5. Conclusion

Both designs provide amicable solutions to maintainability and reusability, with neither providing much more than the other. However, reliability is a key feature for this software solution, due to the use of sensitive data and the nature of the technical requirements. This is the design goal where design B truly shines, and design A fails to deliver. Therefore, the most appropriate selection for this project is:

Design B.