ADR – Architectural Style

**(written independently by Kim Nyaguse)**

**Title:**

Decision of architectural style

**Context + Problem Statement:**

The Advanced Management Library System is a web application that includes features such as profile management, user authentication and wishlist handling which are integrated with Firebase services. It should be able to support scalability to allow for future growth and accessibility for a range of users.

**Decision Driver:**

* A small team with very little experience handling complex systems but with previous experience using firebase.
* The application has many features but isn't complex enough to justify using multiple services.

**Option:**

* Monolith architecture: separate services integrated into a single application.
* Microservices architecture: splits functions into independent services.
* Modular Monolith architecture: functionality organised into distinct modules within a single codebase.
* Service Oriented architecture: loosely coupled services sharing a common communication mechanism.

**Considered Options:**

* Monolith architecture: separate services integrated into a single application.
* Microservices architecture: splits functions into independent services.
* Modular Monolith architecture: functionality organised into distinct modules within a single codebase.

**Decision Outcomes:**

Chosen Option: “Modular Monolith” because it allows for both scalability and simplicity. The functionality can be organised clearly into modules so it can be managed well whilst having all components in a single codebase.

**Consequences:**

* Good because it can be easier developing within a single codebase as a group.
* Good because only one frontend can be a lot easier to implement and test.
* Bad because it doesn't allow for as much scalability as microservices do.
* Bad because depending on a single shared database can become difficult as the application grows.

**Confirmation:**

Can be confirmed by unit and integration tests and ensuring the modules function independently. Architecture reviews would also be beneficial to ensure the implementation aligns with the decision and ensures scalability and accessibility goals are still met.

**Pros and Cons:**

Microservices Architecture

* Good because services can be scaled independently based on demand, which helps to utilise team resources.
* Good because services are separate so if there are any issues with one service, it doesn't affect the entire system.
* Neutral because there's increased autonomy so the team can work without affecting each other.
* Bad because managing multiple services and databases can be extremely difficult, especially for a team that doesn't have a lot of experience.

Traditional Monolithic Architecture

* Good because initial setup, development and deployment is simpler due to it being a single application.
* Good because testing is made easier due to all the functionalities being within a single application.
* Neutral because initial costs are lower due to fewer resources making it more cost-effective.
* Bad because changing from a monolithic to modular or microservices architecture can be extremely difficult.

Template used:

<https://adr.github.io/madr/decisions/adr-template.html>