ADR – Decision of Database

**(written independently by Kim Nyaguse)**

**Title:**

Non-Relational vs Relational Database

**Context + Problem Statement:**

The application requires a database for storing user data, profile information, media inventory etc. The option between a non-relational database and a relational database must align with the project requirements including scalability, accessibility and ease of integration.

**Decision Driver:**

* The database needs to allow for quick prototyping and have integration capabilities.
* The long-term maintenance of the database is an extremely important part of its success.
* The database should be able to support quick data updates coming from multiple clients.

**Option:**

* Non-Relational Database (Firebase, MongoDB)
* Relational Database (MySQL, PostgreSQL)

**Considered Options:**

* Non-Relational Database (Firebase, MongoDB)

**Decision Outcomes:**

Chosen Option: “Firebase (Non-Relational Database)” because of its real-time capabilities and easy integration with front-end frameworks such as React.

**Consequences:**

* Good because Firebase contains a real-time database which aligns with the project requirements of accessibility and a good user experience.
* Good because the costs and complexity are reduced due to it being serverless.
* Bad because it doesn’t support complex queries, and this could limit flexibility in future development phases.

**Confirmation:**

End to end testing can be done on user data integrity, real-time database and scalability. Through code reviews, interaction logic will be reviewed to ensure that it aligns with Firebase’ practices.

**Pros and Cons:**

MongoDB (Non-Relational Database)

* Good because it supports unstructured and dynamic data, which is best for iterative development.
* Good because it has horizontal scaling which supports large-scale applications.
* Neutral because Native real-time synchronization is not as seamless as Firebase.
* Bad because it requires a lot of setup and maintenance compared to other services like Firebase.
* Bad because although it supports complex aggregation queries, advanced aggregation queries need fine-tuning.

MySQL (Relational Database)

* Good because SQL supports complex queries and has clearly defined relationships.
* Good because SQL facilities can easily be transferred to between platforms.
* Bad because SQL requires more effort for optimisation, configuration and optimisation.
* Bad because horizontal scaling can be difficult if there aren't significant architectural changes.

Template used:

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